А.М. Голиков

ШИФРОВАНИЕ В РАДИОЭЛЕКТРОННЫХ СИСТЕМАХ ПЕРЕДАЧИ ИНФОРМАЦИИ

Сборник компьютерных лабораторных работ

Томск 2018

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Томский государственный университет систем управления
и радиоэлектроники

А.М. ГОЛИКОВ

ШИФРОВАНИЕ В РАДИОЭЛЕКТРОННЫХ СИСТЕМАХ ПЕРЕДАЧИ ИНФОРМАЦИИ

Сборник компьютерных лабораторных работ

Томск 2018

УДК 621.39(075.8)

ББК 32.973(я73)

Голиков, А. М. Шифрование в радиоэлектронных системах передачи информации Сборник компьютерных лабораторных работ [Электронный ресурс] / А. М. Голиков. — Томск: ТУСУР, 2018. — 172 с. — Режим доступа: https://edu.tusur.ru/publications/

Сборник компьютерных лабораторных работ предназначен для направления подготовки магистров 11.04.02 "Инфокоммуникационные технологии и системы связи" по магистерским программам подготовки: "Радиоэлектронные системы передачи информации", "Оптические системы связи и обработки информации", "Инфокоммуникационные системы беспроводного широкополосного доспупа", "Защищенные системы связи", для направления подготовки магистров 11.04.01 "Радиотехника" по магистерской программе подготовки: "Радиотехнические системы и комплексы", "Радиоэлектронные устройства передачи информации", "Системы и устройства передачи, приема и обработки сигналов", "Видеоинформационные технологии и цифровое телевидение" и специалитета 11.05.01 "Радиоэлектронные системы и комплексы" специализации "Радиолокационные системы и комплексы", "Радиоэлектронные системы передачи информации", "Радиоэлектронные системы космических комплексов", а также бакалавриата направления 11.03.01 "Радиотехника" (Радиотехнические средства передачи, приема и обработки сигналов), бакалавриата 11.03.02 Инфокоммуникационные технологии и системы связи (Системы мобильной связи, Защищенные системы и сети связи, Системы радиосвязи и радиодоступа, Оптические системы и сети связи) и может быть полезна аспирантам. Представлены описания программных комплексов и методики выполнения лабораторных и практических работ.

СОДЕРЖАНИЕ

Лабораторная работа 1. Исследование классических шифров

Шифры замены

1. Основы шифрования.
2. Шифры однозначной замены.
3. Полиграммные шифры.
4. Омофонические шифры.
5. Полиалфавитные шифры.
6. Нерегулярные шифры.

1. Основы шифрования

Сущность шифрования методом замены заключается в следующем [9]. Пусть шифруются сообщения на русском языке и замене подлежит каждая буква этих сообщений. Тогда, букве **А** исходного алфавита сопоставляется некоторое множество символов (шифрозамен) $\mathbf{M_А}$, **Б –** $\mathbf{M_Б}$, **…, Я –** $\mathbf{M_Я}$. Шифрозамены выбираются таким образом, чтобы любые два множества ($\mathbf{M_I}$ и $\mathbf{M_J}$, $\mathbf{i \neq j}$) не содержали одинаковых элементов ($\mathbf{M_I \cap M_J = Ø}$).

Таблица, приведенная на рис.1.8, является ключом шифра замены. Зная ее, можно осуществить как шифрование, так и расшифрование.

А	Б	…	Я
$M_А$	$M_Б$	…	$M_Я$

Рис. 1.1. Таблица шифрозамен

При шифровании каждая буква **А** открытого сообщения заменяется любым символом из множества $\mathbf{M_А}$. Если в сообщении содержится несколько букв **А**, то каждая из них заменяется на любой символ из $\mathbf{M_А}$. За счет этого с помощью одного ключа можно получить различные варианты шифрограммы для одного и того же открытого сообщения.

Так как множества $\mathbf{M_А}$, $\mathbf{M_Б}$, ..., $\mathbf{M_Я}$ попарно не пересекаются, то по каждому символу шифрограммы можно однозначно определить, какому множеству он принадлежит, и, следовательно, какую букву открытого сообщения он заменяет. Поэтому расшифрование возможно и открытое сообщение определяется единственным образом.

Приведенное выше описание сущности шифров замены относится ко всем их разновидностям за исключением полиалфавитных шифров, в которых для зашифрования разных символов исходного алфавита могут использоваться одинаковые шифрозамены (т.е. $\mathbf{M_I \cap M_J \neq Ø}$, $\mathbf{i \neq j}$).

Метод замены часто реализуется многими пользователями при работе на компьютере. Если по забывчивости не переключить на клавиатуре набор символов с латиницы на

кириллицу, то вместо букв русского алфавита при вводе текста будут печататься буквы латинского алфавита («шифрозамены»).

Шифры замены можно разделить на следующие **подклассы** (разновидности):

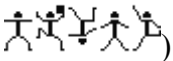
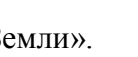
- шифры однозначной замены (моноалфавитные, простые подстановочные). Количество шифрозамен для каждого символа исходного алфавита равно 1 ($|\mathbf{M_i}| = 1$ для одного символа);

- полиграммные шифры. Аналогичен предыдущему за исключением того, что шифрозамене соответствует сразу блок символов исходного сообщения ($|\mathbf{M_i}| = 1$ для блока символов);

- омофонические шифры (однозвучные, многозначной замены). Количество шифрозамен для отдельных символов исходного алфавита больше 1 ($|\mathbf{M_i}| \geq 1$ для одного символа);

- полиалфавитные шифры (многоалфавитные). Состоит из нескольких шифров однозначной замены. Выбор варианта алфавита для зашифрования одного символа зависит от особенностей метода шифрования ($|\mathbf{M_i}| > 1$ для одного символа);

- нерегулярные шифры. Шифрозамены состоят из разного количество символов.

Для записи исходных и зашифрованных сообщений используются строго определенные алфавиты. Алфавиты для записи исходных и зашифрованных сообщений могут отличаться. Символы обоих алфавитов могут быть представлены буквами, их сочетаниями, числами, рисунками, звуками, жестами и т.п. В качестве примера можно привести пляшущих человечков из рассказа А. Конан Дойла () и рукопись рунического письма () из романа Ж. Верна «Путешествие к центру Земли».

2. Шифры однозначной замены

Максимальное количество ключей для любого шифра этого вида не превышает **n!**, где **n** – количество символов в алфавите. С увеличением числа **n** значение **n!** растет очень быстро (1! = 1, 5! = 120, 10! = 3628800, 15! = 1307674368000). При больших **n** для приближенного вычисления **n!** можно воспользоваться формулой Стирлинга

$$n! \approx \sqrt{2\pi n} * \left(\frac{n}{e}\right)^n. \qquad (1.1)$$

Шифр Цезаря. Данный шифр был придуман Гаем Юлием Цезарем и использовался им в своей переписке (1 век до н.э.). Применительно к русскому языку суть его состоит в следующем. Выписывается исходный алфавит (**А, Б, ..., Я**), затем под ним выписывается тот же алфавит, но с циклическим сдвигом на 3 буквы влево.

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Ъ	Э	Ю	Я
Г	Д	Е	Ё	Ж	З	И	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Ъ	Э	Ю	Я	А	Б	В

Рис.1.2. Таблица шифрозамен для шифра Цезаря

При зашифровке буква **А** заменяется буквой **Г**, **Б** - на **Д** и т. д. Так, например, исходное сообщение «АБРАМОВ» после шифрования будет выглядеть «ГДУГПСЕ». Получатель сообщения «ГДУГПСЕ» ищет эти буквы в нижней строке и по буквам над ними восстанавливает исходное сообщение «АБРАМОВ».

Ключом в шифре Цезаря является величина сдвига нижней строки алфавита. Количество ключей для всех модификаций данного шифра применительно к алфавиту русского языка равно 33. Возможны различные модификации шифра Цезаря, в частности лозунговый шифр.

Лозунговый шифр. Для данного шифра построение таблицы шифрозамен основано на лозунге (ключе) – легко запоминаемом слове. Вторая строка таблицы шифрозамен заполняется сначала словом-лозунгом (причем повторяющиеся буквы отбрасываются), а затем остальными буквами, не вошедшие в слово-лозунг, в алфавитном порядке. Например, если выбрано слово-лозунг «ДЯДИНА», то таблица имеет следующий вид.

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Ъ	Э	Ю	Я
Д	Я	И	Н	А	Б	В	Г	Е	Ё	Ж	З	Й	К	Л	М	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Ъ	Э	Ю

Рис. 1.3. Таблица шифрозамен для лозунгового шифра

При шифровании исходного сообщения «АБРАМОВ» по приведенному выше ключу шифрограмма будет выглядеть «ДЯПДКМИ».

В качестве лозунга рекомендуется выбирать фразу, в которой содержаться конечные буквы алфавита. В общем случае, количество вариантов нижней строки (применительно к русскому языку) составляет 33! ($\geq 10^{35}$).

Полибианский квадрат. Шифр изобретен греческим государственным деятелем, полководцем и историком Полибием (III век до н.э.). Применительно к русскому алфавиту суть шифрования заключалась в следующем. В квадрат 6х6 выписываются буквы (необязательно в алфавитном порядке).

	1	2	3	4	5	6
1	А	Б	В	Г	Д	Е
2	Ё	Ж	З	И	Й	К
3	Л	М	Н	О	П	Р
4	С	Т	У	Ф	Х	Ц
5	Ч	Ш	Щ	Ъ	Ы	Ь
6	Э	Ю	Я	-	-	-

Рис. 1.4. Таблица шифрозамен для полибианского квадрата

Шифруемая буква заменяется на координаты квадрата (строка-столбец), в котором она записана. Например, если исходное сообщение «АБРАМОВ», то шифрограмма – «11 12 36 11 32 34 13». В Древней Греции сообщения передавались с помощью оптического телеграфа (с помощью факелов). Для каждой буквы сообщения вначале поднималось количество факелов, соответствующее номеру строки буквы, а затем номеру столбца.

Тюремный шифр. Эта звуковая разновидность полибианского квадрата была разработана заключенными. Система состояла из нескольких ударов, обозначающих строки и столбцы в таблице с буквами алфавита. Один удар, а потом еще два соответствовали строке 1 и столбцу 2, т.е. букве **Б**. Пауза служила разделителем между строками и столбцами. Таким образом, зашифровать исходное сообщение «АБРАМОВ» можно следующим образом.

А	тук ____ тук
Б	тук ____ тук, тук
Р	тук, тук, тук ____ тук, тук, тук, тук, тук, тук
А	тук ____ тук
М	тук, тук, тук ____ тук, тук
О	тук, тук, тук ____ тук, тук, тук, тук
В	тук ____ тук, тук, тук

Рис. 1.5. Пример использования тюремного шифра

Шифрующая система Трисемуса (Тритемия). В 1508 г. аббат из Германии Иоганн Трисемус написал печатную работу по криптологии под названием «Полиграфия». В этой книге он впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра замены обычно использовались таблица для записи букв алфавита и ключевое слово (или фраза). В таблицу сначала вписывалось по строкам ключевое слово, причем повторяющиеся буквы отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку. На рис. 1.6 изображена таблица с ключевым словом «ДЯДИНА».

Д	Я	И	Н	А	Б
В	Г	Е	Ё	Ж	З
Й	К	Л	М	О	П
Р	С	Т	У	Ф	Х
Ц	Ч	Ш	Щ	Ы	Ь
Ъ	Э	Ю	-	-	-

Рис.1.6. Таблица шифрозамен для шифра Трисемуса

Каждая буква открытого сообщения заменяется буквой, расположенной под ней в том же столбце. Если буква находится в последней строке таблицы, то для ее шифрования берут самую верхнюю букву столбца. Например, исходное сообщение «АБРАМОВ», зашифрованное – «ЖЗЦЖУФЙ».

Шифр масонов. В XVIII в. масоны создали шифр, чтобы скрыть от общественности свои коммерческие сделки. Как поведали те, кто прежде состоял в рядах этого общества, масоны пользовались способом засекречивания, весьма похожим на шифр розенкрейцеров. В «решетке» и в углах находятся точки, которыми заменяются буквы:

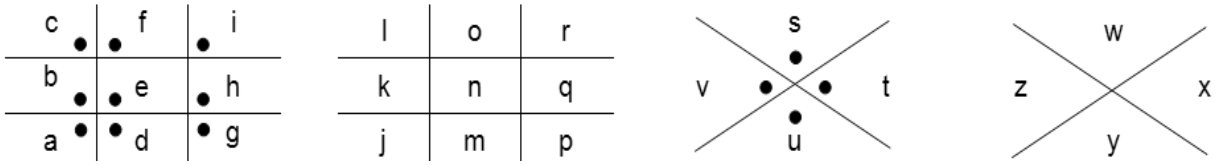


Рис. 1.7. Шифр масонов

Так как клятвы хранить тайну нарушались не раз, большинство Великих лож масонов в США больше не пользуются письменными шифрами, предпочитая передавать устные инструкции во время закрытых ритуалов.

С помощью шифра масонов можно легко расшифровать следующую фразу.

Рис. 1.8. Пример использования шифра масонов

Это первый уровень, на котором находятся все впервые вступившие в общество члены: Blue Lodge (рус. «Голубая (Синяя) ложа»).

Одним из существенных **недостатков шифров однозначной замены** является их легкая вскрываемость. При вскрытии шифрограмм используются различные приемы, которые даже при отсутствии мощных вычислительных средств позволяют добиться положительного результата. Один из таких приемов базируется на том, что в шифрограммах остается информация о частоте встречаемости букв исходного текста. Если в открытом сообщении часто встречается какая-либо буква, то в шифрованном сообщении также часто будет встречаться соответствующий ей символ. Еще в 1412 году Шихаба ал-Калкашанди в своем труде «Субх ал-Ааша» привел таблицу частоты появления арабских букв в тексте на основе анализа текста Корана. Для разных языков мира существуют подобные таблицы. Так, например, для русского языка такая таблица выглядит следующим образом.

Таблица 1.1 Вероятности появления букв русского языка в текстах*

Буква (символ)	Вероят-ность	Буква	Вероят-ность	Буква	Вероят-ность	Буква	Вероят-ность
Пробел	0.146	Р	0.042	Я	0.017	Ж	0.007
О	0.094	Л	0.039	З	0.016	Ш	0.006
Е	0.071	В	0.038	Ы	0.015	Ц, Ю	0.005
А	0.069	К	0.029	Г	0.014	Щ	0.004
И	0.064	М	0.027	Ь, Б	0.013	Ф	0.003
Н	0.057	П	0.026	Ч	0.012	Э	0.002
Т	0.054	Д	0.024	Й	0.010	Ъ	0.001
С	0.046	У	0.023	Х	0.008		

*) В таблице приведены оценки вероятностей появления букв русского языка и пробела, полученные на основе анализа научно-технических и художественных текстов общим объемом более 1000000 символов.

Существуют подобные таблицы для пар букв (биграмм). Например, часто встречаемыми биграммами являются «то», «но», «ст», «по», «ен» и т.д. Другой прием вскрытия шифрограмм основан на исключении возможных сочетаний букв. Например, в текстах (если они написаны без орфографических ошибок) нельзя встретить сочетания «чя», «щы», «ьъ» и т.п.

Для усложнения задачи вскрытия шифров однозначной замены еще в древности перед шифрованием из исходных сообщений исключали пробелы и/или гласные буквы. Другим способом, затрудняющим вскрытие, является шифрование **биграммами** (парами букв).

Полиграммные шифры

Полиграммные шифры замены - это шифры, в которых одна шифрозамена соответствует сразу нескольким символам исходного текста.

Биграммный шифр Порты. Шифр Порты, представленный им в виде таблицы, является первым известным биграммным шифром. Размер его таблицы составлял 20 x 20 ячеек; наверху горизонтально и слева вертикально записывался стандартный алфавит (в нем не было букв J, K, U, W, X и Z). В ячейках таблицы могли быть записаны любые числа, буквы или символы - сам Джованни Порта пользовался символами - при условии, что содержимое ни одной из ячеек не повторялось. Применительно к русскому языку таблица шифрозамен может выглядеть следующим образом.

	А	Б	В	Г	Д	Е (Ё)	Ж	З	И (Й)	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
А	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
Б	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062
В	063	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079	080	081	082	083	084	085	086	087	088	089	090	091	092	093
Г	094	095	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124
Д	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155
Е (Ё)	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186
Ж	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217
З	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248
И (Й)	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279
К	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310
Л	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341
М	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372
Н	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403
О	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434
П	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465
Р	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496
С	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527
Т	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558
У	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589
Ф	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620
Х	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651
Ц	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682
Ч	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713
Ш	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744
Щ	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775
Ъ	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806
Ы	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837
Ь	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868
Э	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899
Ю	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930
Я	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961

Рис. 1.9. Таблица шифрозамен для шифра Порты

Шифрование выполняется парами букв исходного сообщения. Первая буква пары указывает на строку шифрозамены, вторая - на столбец. В случае нечетного количества букв в исходном сообщении к нему добавляется вспомогательный символ («пустой знак»). Например, исходное сообщение «АБ РА МО В», зашифрованное – «002 466 355 093». В качестве вспомогательного символа использована буква «Я».

Шифр Playfair (англ. «Честная игра»). В начале 1850-х гг. Чарлз Уитстон придумал так называемый «прямоугольный шифр». Леон Плейфер, близкий друг Уитстона, рассказал об этом шифре во время официального обеда в 1854 г. министру внутренних дел лорду Пальмерстону и принцу Альберту. А поскольку Плейфер был хорошо известен в военных и дипломатических кругах, то за творением Уитстона навечно закрепилось название «шифр Плейфера».

Данный шифр стал первым буквенным биграммным шифром (в биграммной таблице Порты использовались символы, а не буквы). Он был предназначен для обеспечения секретности телеграфной связи и применялся британскими войсками в Англо-бурской и Первой мировой войнах. Им пользовалась также австралийская служба береговой охраны островов во время Второй мировой войны.

Шифр предусматривает шифрование пар символов (биграмм). Таким образом, этот шифр более устойчив к взлому по сравнению с шифром простой замены, так как затрудняется частотный анализ. Он может быть проведен, но не для 26 возможных символов (латинский алфавит), а для 26 х 26 = 676 возможных биграмм. Анализ частоты биграмм возможен, но является значительно более трудным и требует намного большего объема зашифрованного текста.

Для шифрования сообщения необходимо разбить его на биграммы (группы из двух символов), при этом, если в биграмме встретятся два одинаковых символа, то между ними добавляется заранее оговоренный вспомогательный символ (в оригинале – **X**, для русского алфавита - **Я**). Например, «зашифрованное сообщение» становится «за ши фр ов ан но ес о**Я** об ще ни е**Я**». Для формирования ключевой таблицы выбирается лозунг и далее она заполняется по правилам шифрующей системы Трисемуса. Например, лозунг «ДЯДИНА»

Д	Я	И	Н	А	Б
В	Г	Е	Ё	Ж	З
Й	К	Л	М	О	П
Р	С	Т	У	Ф	Х
Ц	Ч	Ш	Щ	Ы	Ь
Ъ	Э	Ю	-	1	2

Рис.1.10. Ключевая таблица для шифра Playfair

Затем, руководствуясь следующими правилами, выполняется зашифровывание пар символов исходного текста:

1. Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.

2. Если символы биграммы исходного текста встречаются в одном столбце, то они преобразуются в символы того же столбца, находящимися непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.

3. Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

Пример шифрования.

- биграмма «за» формирует прямоугольник – заменяется на «жб»;
- биграмма «ши» находятся в одном столбце – заменяется на «юе»;
- биграмма «фр» находятся в одной строке – заменяется на «хс»;
- биграмма «ов» формирует прямоугольник – заменяется на «йж»;
- биграмма «ан» находятся в одной строке – заменяется на «ба»;
- биграмма «но» формирует прямоугольник – заменяется на «ам»;
- биграмма «ес» формирует прямоугольник – заменяется на «гт»;
- биграмма «оя» формирует прямоугольник – заменяется на «ка»;
- биграмма «об» формирует прямоугольник – заменяется на «па»;
- биграмма «ще» формирует прямоугольник – заменяется на «шё»;
- биграмма «ни» формирует прямоугольник – заменяется на «ан»;
- биграмма «ея» формирует прямоугольник – заменяется на «ги».

Шифрограмма – «жб юе хс йж ба ам гт ка па шё ан ги».

Для расшифровки необходимо использовать инверсию этих правил, откидывая символы **Я** (или **Х**), если они не несут смысла в исходном сообщении.

Шифр Хилла. Первый практически реализуемый способ шифрования с использованием алгебры был придуман в 1929 г. математиком Лестером Хиллом - профессором из Хантер-колледжа в Нью-Йорке, статья которого «Cryptography in an Algebraic Alphabet» была опубликована в журнале «The American Mathematical Monthly».

Каждой букве алфавита сопоставляется число. Для русского алфавита можно использовать простейшую схему: А = 0, Б = 1, ..., Я = 32. Для зашифрования блок исходного сообщения из **n** букв рассматривается как n-мерный вектор чисел и умножается на матрицу размером **n** x **n** по модулю 33. Данная матрица, совместно с кодовой таблицей сопоставления букв алфавита с числами, является ключом зашифрования. Для расшифрования применяется обратная матрица[1] по модулю.

Например, для триграммных замен могут использоваться следующие матрицы зашифрования / расшифрования.

6	27	1
13	16	32
28	17	15

матрица зашифрования

2	26	17
26	20	4
13	30	21

матрица расшифрования

Рис. 1.11 Матрицы зашифрования / расшифрования

Исходное сообщение «АБРАМОВ», дополненное двумя вспомогательными буквами «яя» (для кратности трем), после сопоставления букв с числами будет выглядеть следующим образом «0 1 17 0 13 15 2 32 32». После перемножения троек чисел на матрицу зашифрования шифрограмма примет следующий вид «11 32 8 3 28 17 17 11 24» (или в буквенном эквиваленте «КЯЗ ГЬР РКЧ»).

АБР - 0 1 17

(6 * 0 + 27 * 1 + 1 * 17) mod 33 = 11 (К)

(13 * 0 + 16 * 1 + 32 * 17) mod 33 = 32 (Я)

(18 * 0 + 17 * 1 + 15 * 17) mod 33 = 8 (З)

АМО - 0 13 15

(6 * 0 + 27 * 13 + 1 * 15) mod 33 = 3 (Г)

(13 * 0 + 16 * 13 + 32 * 15) mod 33 = 28 (Ь)

(28 * 0 + 17 * 13 + 15 * 15) mod 33 = 17 (Р)

Вяя - 2 32 32

(6 * 2 + 27 * 32 + 1 * 32) mod 33 = 17 (Р)

(13 * 2 + 16 * 32 + 32 * 32) mod 33 = 11 (К)

(28 * 2 + 17 * 32 + 15 * 32) mod 33 = 24 (Ч)

Для расшифрования тройки чисел шифрограммы необходимо умножить на матрицу расшифрования.

КЯЗ - 11 32 8

(2 * 11 + 26 * 32 + 17 * 8) mod 33 = 0 (А)

(26 * 11 + 20 * 32 + 4 * 8) mod 33

= 1 (Б)

(13 * 11 + 30 * 32 + 21 * 8) mod 33 = 17 (Р)

ГЬР - 3 28 17

(2 * 3 + 26 * 28 + 17 * 17) mod 33 = 0 (А)

(26 * 3 + 20 * 28 + 4 * 17) mod 33 = 13 (М)

(13 * 3 + 30 * 28 + 21 * 17) mod 33 = 15 (О)

РКЧ - 17 11 24

(2 * 17 + 26 * 11 + 17 * 24) mod 33 = 2 (В)

(26 * 17 + 20 * 11 + 4 * 24) mod 33 = 32 (я)

(13 * 17 + 30 * 11 + 21 * 24) mod 33 = 32 (я)

В результате будет получен набор чисел «0 1 17 0 13 15 2 32 32», соответствующий исходному сообщению со вспомогательными символами «АБРАМОВяя».

[1]**Обратная матрица** - матрица A^{-1}, при умножении на которую, исходная матрица А дает в результате единичную матрицу Е.

Омофонические шифры

Другое направление повышения стойкости шифров замены состоит в том, чтобы каждое множество шифрообозначений $\mathbf{M_i}$ содержало более одного элемента. При использовании такого шифра одну и ту же букву (если она встречается несколько раз в сообщении) заменяют на разные шифрозамены из $\mathbf{M_i}$. Это позволяет скрыть истинную частоту встречаемости букв открытого сообщения.

Система омофонов. В 1401 г. Симеоне де Крема стал использовать таблицы омофонов для сокрытия частоты появления гласных букв в тексте при помощи более чем одной шифрозамены. Такие шифры позже стали называться **шифрами многозначной замены** или **омофонами**[2]. Они получили развитие в XV веке. В книге «Трактат о шифрах» Леона Баттисты Альберти (итальянский ученый, архитектор, теоретик искусства, секретарь папы Климентия XII), опубликованной в 1466 г., приводится описание шифра замены, в

котором каждой букве ставится в соответствие несколько эквивалентов, число которых пропорционально частоте встречаемости буквы в открытом тексте. Так, если ориентироваться на то число шифрозамен для буквы **О** должно составлять 94, для буквы **Е** – 71 и т.д. При этом каждая шифрозамена должна состоять из 3 цифр и их общее количество равно 1000. На рис.1.12 представлен фрагмент таблицы шифрозамен.

№ п/п	Пробел	А	Б	В	…	М	…	О	…	Р	…	Я
1	012	311	128	175	…	037	…	248	…	064	…	266
2	042	357	950	194	…	149	…	267	…	189	…	333
…	…	…	…	…	…	…	…	…	…	…	…	…
13	278	495	990	199	…	349	…	303	…	374	…	749
…	…	…		…	…	…	…	…	…	…	…	…
17	342	519		427	…	760	…	306	…	469	…	845
…	…	…		…	…	…	…	…	…	…		
27	437	637		524	…	777	…	432	…	554		
…	…	…		…				…	…	…		
38	457	678		644				824	…	721		
…	…	…						…	…	…		
42	628	776						828	…	954		
…	…	…						…				
69	681	901						886				
…	…							…				
94	974							903				
…	…											
146	976											

Рис.1.12. Фрагмент таблицы шифрозамен для системы омофонов

При шифровании символ исходного сообщения заменяется на любую шифрозамену из своего столбца. Если символ встречается повторно, то, как правило, используют разные шифрозамены. Например, исходное сообщение «АБРАМОВ» после шифрования может выглядеть «357 990 374 678 037 828 175».

Книжный шифр. Заметным вкладом греческого ученого Энея Тактика в криптографию является предложенный им так называемый книжный шифр, описанный в сочинении «Об обороне укреплённых мест». Эней предложил прокалывать малозаметные дырки в книге или в другом документе над буквами секретного сообщения. Интересно отметить, что в первой мировой войне германские шпионы использовали аналогичный шифр, заменив дырки на точки, наносимые симпатическими чернилами[3] на буквы газетного текста.

После первой мировой войны книжный шифр приобрел иной вид. Шифрозамена для каждой буквы определялась набором цифр, которые указывали на номер страницы, строки и позиции в строке. Количество книг, изданных за всю историю человечества, является величиной ограниченной (по крайней мере, явно меньше, чем 15!). Однако отсутствие полной электронной базы по изданиям делает процедуру вскрытия шифрограмм почти не выполнимой. В связи с этим книжный шифр относят к категории совершенных.

Вариантные шифры . Вариантные шифры напоминают полибианский квадрат, но для каждой строки и столбца используется по два буквенных идентификатора. В квадрат

(прямоугольник) шифрозамен вначале записывается ключевое слово без повторяющихся букв, а затем дополняется не вошедшими в него буквами по порядку следования в алфавите. Каждой строке и столбцу квадрата ставится в соответствие по две буквы алфавита. Буквы для идентификации строк и столбцов не должны повторяться.

	Й Ц	У К	Е Н	Г Ш	Щ З	Х Ъ
Ф Ы	Д	Я	И	Н	А	Б
В А	В	Г	Е	Ё	Ж	З
П Р	Й	К	Л	М	О	П
О Л	Р	С	Т	У	Ф	Х
Д Ж	Ц	Ч	Ш	Щ	Ы	Ь
Э Я	Ъ	Э	Ю	-	-	-

Рис. 1.13. Пример таблицы шифрозамен вариантного шифра с ключевым словом «ДЯДИНА»

Комбинации букв-идентификаторов строки и столбца дают по восемь шифрозамен для каждой буквы исходного текста. Например, для буквы **Д** возможны шифрозамены: **ФЙ**,**ЙФ**, **ФЦ**, **ЦФ**, **ЫЙ**, **ЙЫ**, **ЫЦ** и **ЦЫ**. Для таблицы шифрозамен, приведенной исходное сообщение «АБРАМОВ» может быть зашифровано как «ЫЗ ЫХ ОЦ ЗФ ГР РЩ АЙ».

---

[2]**Омофоны** (греч. homos - одинаковый и phone - звук) - слова, которые звучат одинаково, но пишутся по-разному и имеют разное значение.

[3]**Симпатические (невидимые) чернила** — чернила, записи которыми являются изначально невидимыми и становятся видимыми только при определенных условиях (нагрев, освещение, химический проявитель и т. д.).

Полиалфавитные шифры

Напомним, что полиалфавитные шифры состоят из нескольких шифров однозначной замены и отличаются друг от друга способом выбор варианта алфавита для зашифрования одного символа.

Диск Альберти. В «Трактате о шифрах» Альберти приводит первое точное описание многоалфавитного шифра на основе шифровального диска.

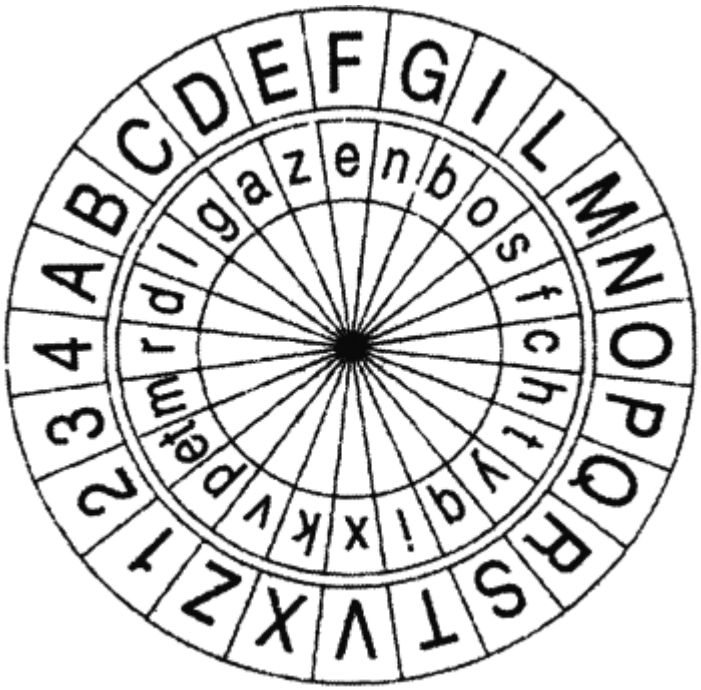


Рис.1.14. Диск Альберти

Он состоял из двух дисков – внешнего неподвижного (на нем были нанесены буквы в алфавитном порядке и цифры 1, 2, 3, 4) и подвижного внутреннего диска на котором буквы были переставлены. Процесс шифрования заключался в нахождении буквы открытого текста на внешнем диске и замене ее на букву с внутреннего диска, стоящую под ней. После этого внутренний диск сдвигался на одну позицию и шифрование второй буквы производилось уже по новому шифралфавиту. Ключом данного шифра являлся порядок расположения букв на внутреннем диске и его начальное положение относительно внешнего диска.

Таблица Трисемуса. Одним из шифров, придуманных немецким аббатом Трисемусом, стал многоалфавитный шифр, основанный на так называемой «таблице Трисемуса» - таблице со стороной равной **n**, где **n** – количество символов в алфавите. В первой строке матрицы записываются буквы в порядке их очередности в алфавите, во второй – та же последовательность букв, но с циклическим сдвигом на одну позицию влево, в третьей – с циклическим сдвигом на две позиции влево и т.д.

А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Рис.1.15. Таблица Трисемуса

Здесь первая строка является одновременно и строкой букв открытого текста. Первая буква текста шифруется по первой строке, вторая буква по второй и так далее после использования последней строки вновь возвращаются к первой. Так сообщение «АБРАМОВ» приобретет вид «АВТГРУИ».

Система шифрования Виженера. В 1586 г. французский дипломат Блез Виженер представил перед комиссией Генриха III описание простого, но довольно стойкого шифра, в основе которого лежит таблица Трисемуса.

Перед шифрованием выбирается ключ из символов алфавита. Сама процедура шифрования заключается в следующем. По i-ому символу открытого сообщения в первой строке определяется столбец, а по i-ому символу ключа в крайнем левом столбце – строка. На пересечении строки и столбца будет находиться i-ый символ, помещаемый в шифрограмму. Если длина ключа меньше сообщения, то он используется повторно. Например, исходное сообщение «АБРАМОВ», ключ – «ДЯДИНА», шифрограмма – «ДАФИЩОЖ».

Справедливости ради, следует отметить, что авторство данного шифра принадлежит итальянцу Джованни Батиста Беллазо, который описал его в 1553 г. История «проигнорировала важный факт и назвала шифр именем Виженера, несмотря на то, что он ничего не сделал для его создания». Беллазо предложил называть секретное слово или фразу **паролем** (ит. password; фр. parole - слово).

В 1863 г. Фридрих Касиски опубликовал алгоритм атаки на этот шифр, хотя известны случаи его взлома шифра некоторыми опытными криптоаналитиками ещё в XVI веке. Несмотря на это шифр Виженера имел репутацию исключительно стойкого к «ручному» взлому еще долгое время. Так, известный писатель и математик Чарльз Лютвидж Доджсон (Льюис Кэрролл) в своей статье «Алфавитный шифр», опубликованной в детском журнале в 1868 г., назвал шифр Виженера невзламываемым. В 1917 году научно-популярный журнал «Scientific American» также отозвался о шифре Виженера, как о неподдающемся взлому.

Роторные машины. Идеи Альберти и Беллазо использовались при создании электромеханических роторных машин первой половины XX века. Некоторые из них использовались в разных странах вплоть до 1980-х годов. Большинство использовало понятие ротора - механического колеса, используемого для выполнения подстановки. Наиболее известной из роторных машин является немецкая машина времен Второй мировой войны «Энигма».

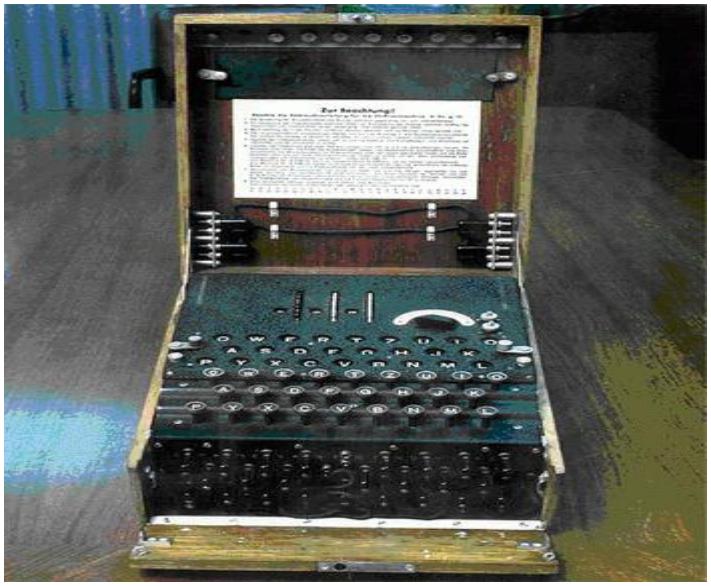


Рис. 1.16. Шифровальная машина Энигма

Роторная машина, включающая клавиатуру и набор роторов, реализует вариант шифра Виженера. Каждый ротор представляет собой произвольное размещение алфавита, имеет 26 позиций (применительно к латинскому алфавиту) и выполняет простую подстановку. Например, ротор может быть использован для замены **A** на **F**, **B** на **U**, **C** на **I** и так далее.

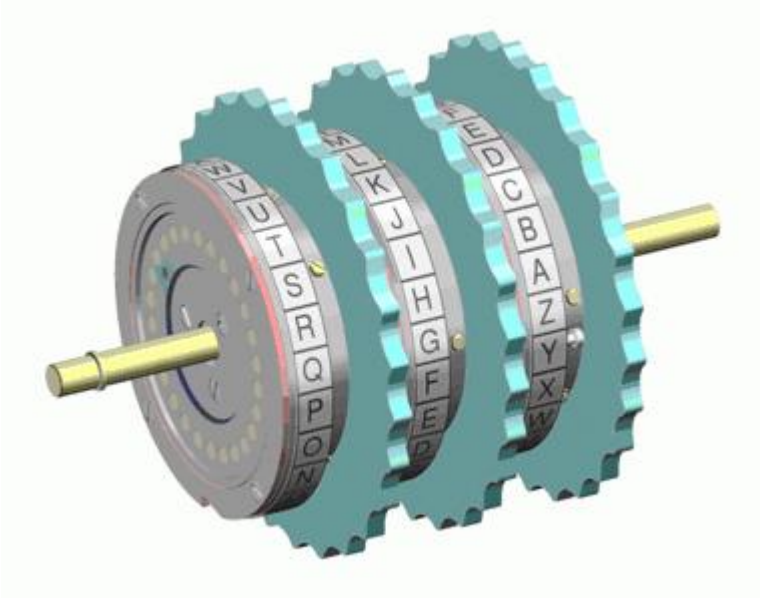


Рис. 1.17. Три последовательно соединённых ротора

Выходные штыри одного ротора соединены с входными штырями следующего ротора и при нажатии символа исходного сообщения на клавиатуре замыкали электрическую цепь, в результате чего загоралась лампочка с символом шифрозамены.

Шифрующее действие «Энигмы» показано для двух последовательно нажатых клавиш - ток течёт через роторы, «отражается» от рефлектора, затем снова через роторы.

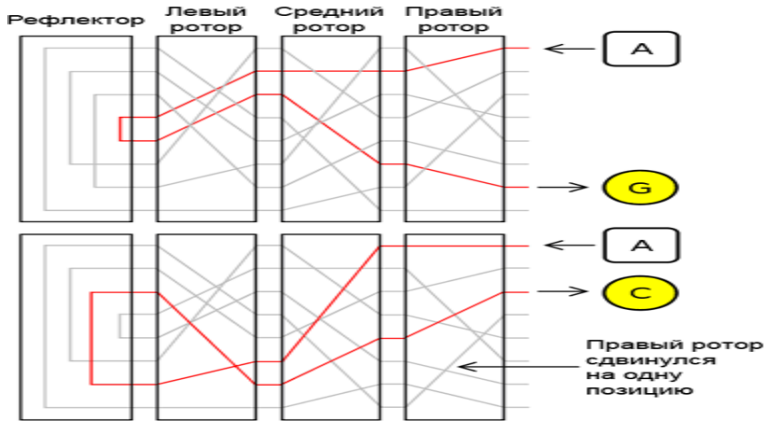


Рис. 1.18. Схема шифрования

Примечание. Серыми линиями показаны другие возможные электрические цепи внутри каждого ротора. Буква **A** шифруется по-разному при последовательных нажатиях одной клавиши, сначала в **G**, затем в **C**. Сигнал идет по другому маршруту за счёт поворота ротора после нажатия предыдущей буквы исходного сообщения.

В некоторых шифрах в самой шифрограмме могут содержаться символы, предписывающие использование того или иного алфавита.

Шифры Тени. Главными развлечениями для американцев тридцатых годов XX века были бульварное чтиво и радио. Для раскрутки своих книжек издательство Street & Smith проспонсировало радиопередачу, ведущим в которой был Тень (англ. Shadow), загадочный

рассказчик со зловещим голосом, который в начале каждого выпуска заявлял: «Кто знает, что за зло прячется в сердцах людей? Тень знает!». Успех радиопередачи подтолкнул издательство к решению начать выпускать серию книг, в которой главным героем был бы Тень. Свои услуги предложил Уолтер Гибсон, большой любитель фокусов и головоломок. Под псевдонимом Максвелл Грант он принялся писать роман за романом, да с такой скоростью, что за свою жизнь написал почти 300 книжек о грозе тех, кто нечист помыслами. В новелле «Цепочка смерти» супергерой воспользовался так называемым кодом направления, хотя на самом деле он действует скорее как шифр, чем как код:

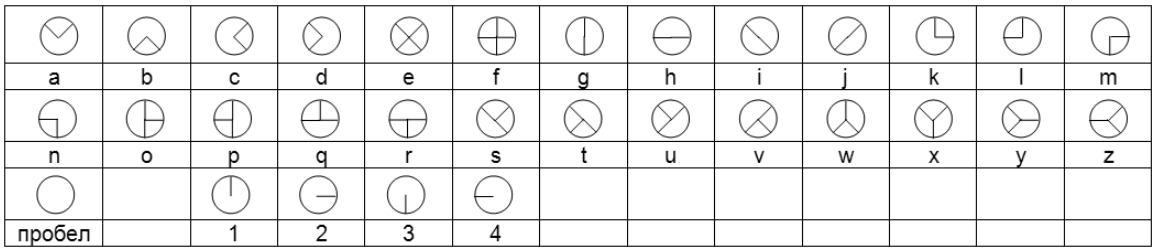


Рис. 1.19. Таблица шифрозамен и управляющих символов

Управляющие символы в последней строке таблицы служат для изменения кода (выбора шифралфавита) для зашифрования/дешифрования. Линии внутри каждого кружка фактически являются стрелками, подсказывающими адресату, как держать лист бумаги. Символ 1 означает, что лист надо держать как обычно: верх и низ расположены на своих местах, а сообщение читается слева направо. Символ 2 требует поворота на 90° вправо, а символ 3 указывает, что лист бумаги следует перевернуть вверх ногами. Символ 4 обозначает поворот на 90° влево.

Эти дополнительные символы могут появляться перед любой строчкой текста, а также в ее середине.

Из нижеприведенного примера можно узнать настоящие имя и фамилию супергероя.

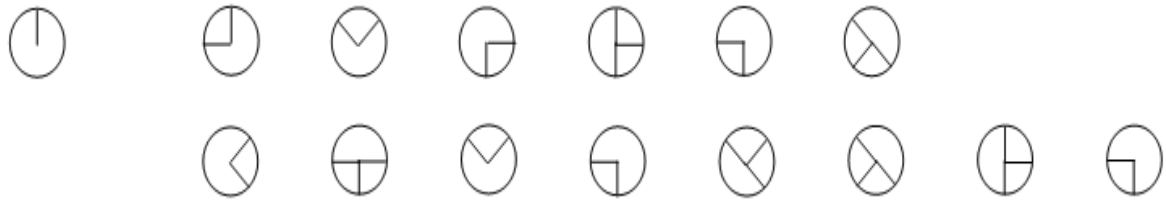


Рис. 1.20. Настоящие имя и фамилия Тени

Согласно первому управляющему символу, лист следует держать обычным образом, не поворачивая, и после замены буквы образуют «Lamont Cranston» (Ламонт Крэнстон).

Нерегулярные шифры

Еще одним направлением повышения стойкости шифров замены заключается в использовании нерегулярных шифров. В приведенных выше шифрах (**регулярных**) шифрозамены состоят из строго определенного количества символов (букв, цифр,

графических элементов и т.д.) или в шифрограмме они отделяются друг от друга специальными символами (пробелом, точкой, запятой и т.д.). В нерегулярных шифрах шифрозамены состоят из разного количества символов и записываются в шифрограмме в подряд (без выделения друг от друга), что значительно затрудняет криптоанализ.

Совмещенный шифр (совмещенная таблица). Данный шифр применялся еще семейством Ардженти - криптологами, разрабатывавшими шифры для Папы Римского в XVI в. В XX столетии этим способом пользовались коммунисты в ходе гражданской войны в Испании. В начале войны противники фашизма в Испании контролировали большинство крупных городов и защищали свою связь, включая радиопередачи, с помощью различных методов шифрования, в том числе совмещенных шифров.

Вариант коммунистов получил название «совмещенный» из-за необычного использования одно- и двухцифровых шифрозамен, благодаря чему сообщение приобретало дополнительную защиту от потенциального дешифровальщика. Некоторые буквы зашифровывались одной цифрой, другие же - парой цифр. При этом криптоаналитик противника совершенно не представлял, где в перехваченных сообщениях находятся одноцифровые, а где двухцифровые шифрозамены.

Таблица шифрозамен состоит из 10 столбцов с нумерацией 0, 9, 8, 7, 6, 5, 4, 3, 2 и 1. В начальную строку вписывается ключевое слово без повторяющихся букв. В последующие строки вписываются по десять не вошедших в него букв по порядку следования в алфавите. Строки, за исключением начальной, нумеруются по порядку, начиная с 1.

	0	9	8	7	6	5	4	3	2	1
	Д	Я	И	Н	А					
1	Б	В	Г	Е	Ё	Ж	З	Й	К	Л
2	М	О	П	Р	С	Т	У	Ф	Х	Ц
3	Ч	Ш	Щ	Ы	Ь	Ъ	Э	Ю	-	-

Рис. 1.21. Пример таблицы шифрозамен совмещенного шифра с ключевым словом «ДЯДИНА»

При шифровании буквы исходного сообщения, входящие в ключевое слово, заменяются на одну цифру (номер столбца), остальные – двумя (номера строки и столбца). Например, для приведенной выше таблицы шифрозамен исходное сообщение «АБРАМОВ» будет зашифровано как «610276202919».

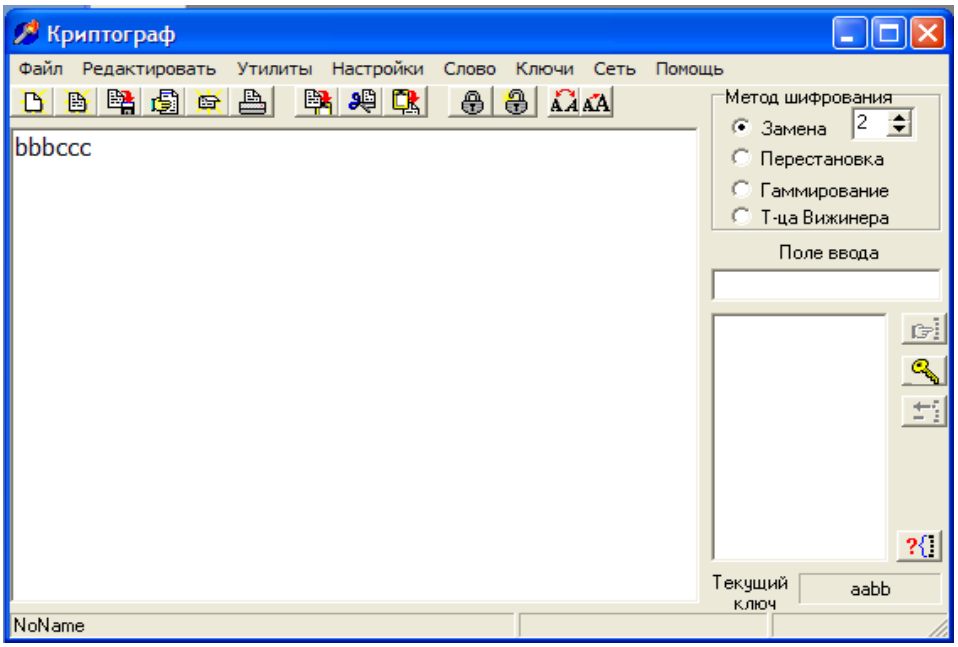
При получении шифрограммы адресат знает, что когда появляются цифры 1, 2 или 3, с ними обязательно связана еще одна цифра, поскольку они представляют собой цифровую пару. Так что 35 - это, несомненно, пара, а 53 - нет, ведь в таблице нет строки с номером 5. Перехват такого сообщения третьей стороной даст ей всего лишь ряд цифр, потому что

криптоаналитик противника не имеет ни малейшего представления, какие цифры одиночные, а какие входят в состав пар.

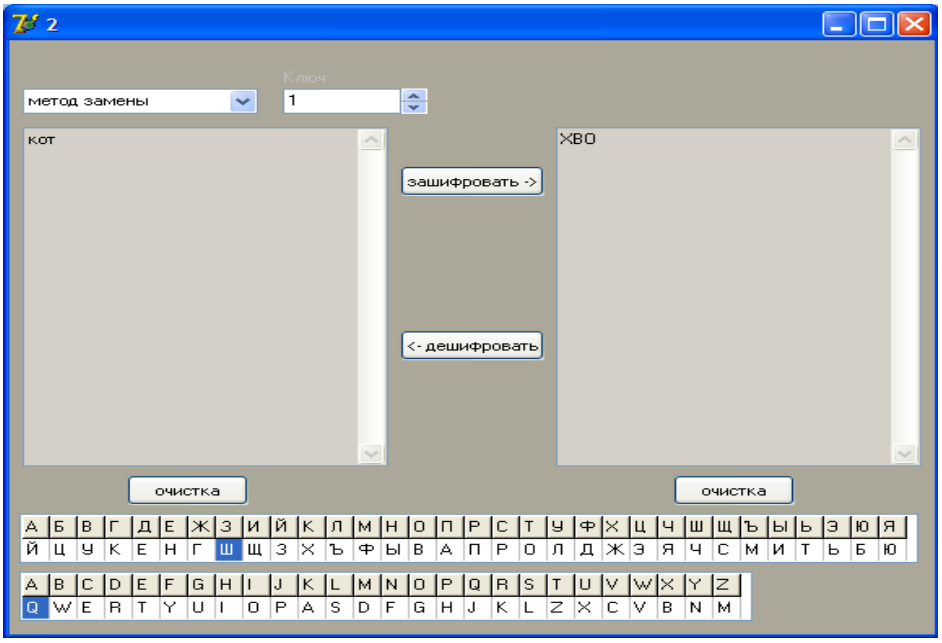
Лабораторный комплекс

В программе реализуется шифр замены, перестановки, гаммирования и шифрование на основе таблицы Вижинера. В поле метод шифрования можно выбрать ключ, на который сдвигается шифруемое сообщение. Специальными кнопками можно выбрать шифрование или дешифрование.

Задание 1.

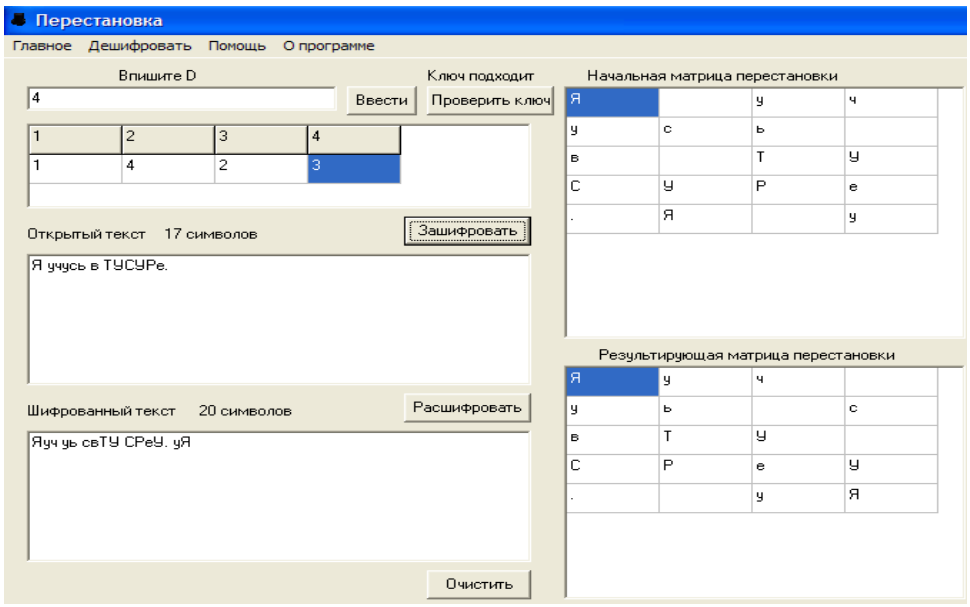


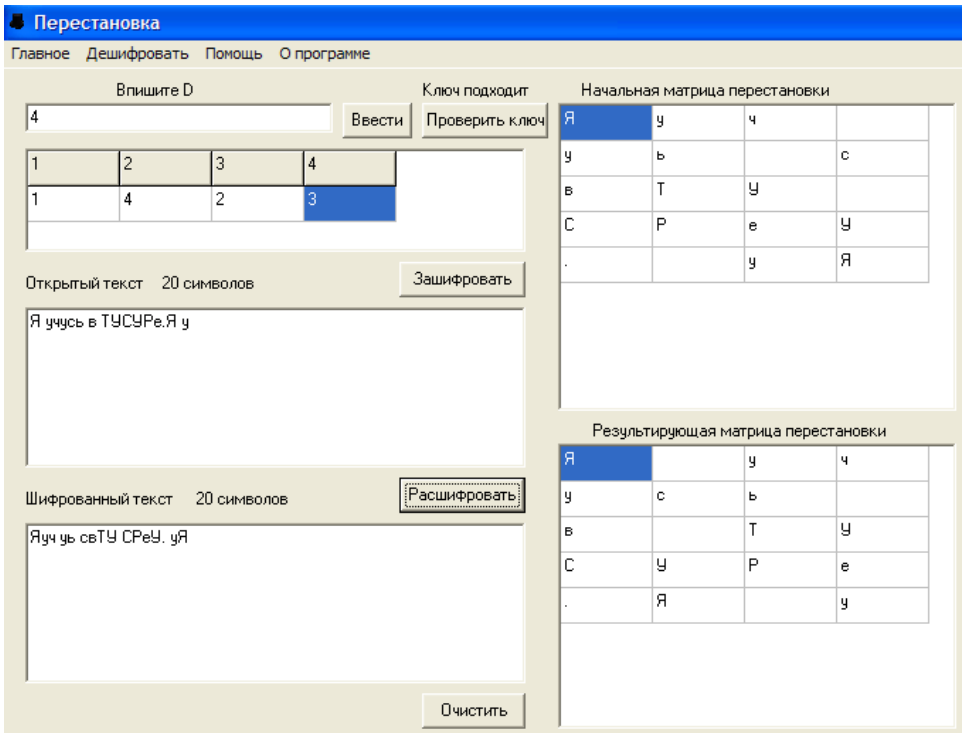
Задание 2.



В данной программе шифр замены имеет специальную таблицу для шифрования, поэтому ключом является таблица.

Задание 3





Лабораторная работа 2. Криптоанализ шифров гаммирования

Целью работы является получение практических навыков криптоанализа аддитивных двоичных шифров методом вероятных слов. Для того, чтобы не загромождать основную цель рутинными действиями в данной работе были сделаны некоторые упрощения по сравнению с реальными ситуациями. Такими упрощениями являются:

- Заранее известная длина регистра
- Короткое сообщение
- Выбор только из наиболее вероятных биграмм (8 штук)

Задание на лабораторную работу

Заданием для данной лабораторной работы является отыскание открытого текса зашифрованного методом гаммирования при помощи сдвигового регистра с линейной обратной связью. Для сдачи работы необходимо предоставить текст файла отчета. **После получения верного открытого текста необходимо по найденной части ключа вручную определить положение отводов в регистре при помощи алгоритма Берлекэмпа-Месси** и представить таблицу вывода для проверки. Необходимо заметить, что это является обязательным шагом уже после нахождения верного открытого текста. Для промежуточных нахождений положений отводов в регистре алгоритм Берлекэмпа-Месси использовать необязательно, можно воспользоваться методом, основанным на нахождении обратной матрицы.

Общее описание лабораторной работы

Целью работы является приобретение практических навыков криптоанализа аддитивных шифров.

Результатом работы является получение осмысленного открытого текста из зашифрованного сообщения при помощи учебной программы, называемой «Криптоанализ аддитивного шифра LSR». Лабораторная работа представляет собой исполняемый файл LSR.exe (учебная программа) и набор из 25 вариантов задания (зашифрованный текст).

Общий вид окна учебной программы

Программа LSR.exe представляет собой исполняемый файл, который запускается двойным нажатием на пиктограмму

Рис 1.22. Пиктограмма LSR

После чего на экране появляется диалоговое окно, представляющее собой окно лабораторной работы (рабочее окно).

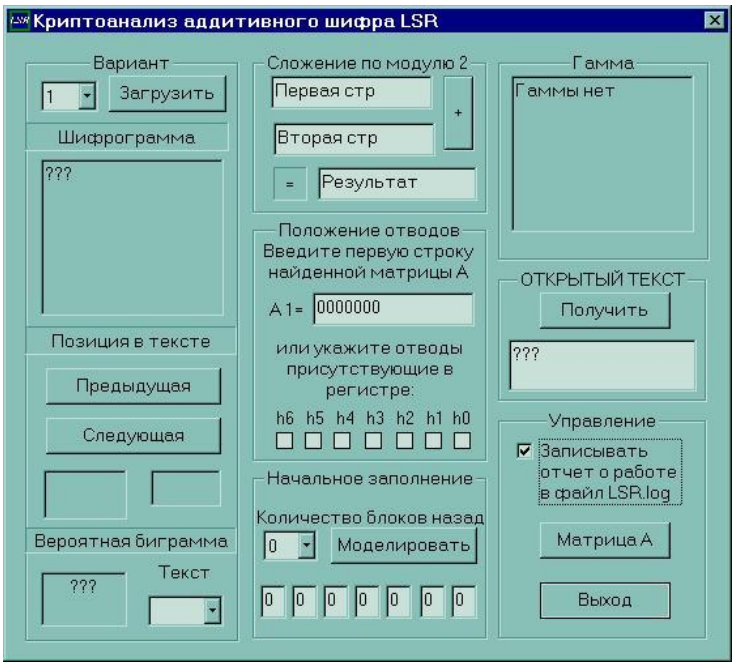


Рис 1.23. Внешний вид окна LSR

Рабочее окно лабораторной работы разделено на 7 блоков, которые представляют собой отдельно последовательно выполняемые шаги лабораторной работы и блок управления. Работа последующих блоков базируется на результатах работы предыдущих.

Кратко перечислим и поясним эти блоки:

- Вариант. Блок предназначен для загрузки внешнего файла варианта в соответствии с выбранным номером, отображения текста задания (подзаголовок Шифрограмма) в зашифрованном виде в битовом представлении и выбора одной из наиболее вероятных биграмм (подзаголовок Вероятная биграмма). Кроме того в блоке находятся кнопки управления «Предыдущая» и «Следующая» для перехода к соответственно предыдущий и последующей позиции, которая является вероятной позицией для биграммы;
- Сложение по модулю два. Блок предназначен для отыскания части вероятной гаммы путем сложения по модулю два битового представления вероятной биграммы и битового представления выбранной части зашифрованного текста;
- Положение отводов. Блок предназначен для ввода строки матрицы А, которая определяет положение отводов в регистре, или указания положения отводов путем заполнения соответствующих полей. **Положение отводов определяется студентом используя подпрограмму**, которая вызывается нажатием кнопки «Матрица А». **Выполняющий составляет вектора S(1),…,S(8)** и подпрограмма, используя метод основанный на нахождении обратной матрицы (с помощью метода Гаусса), находит матрицу обратную к X1 и матрицу А (значение первой строки, которой необходимо для определения положения отводов).

- Начальное заполнение. Блок предназначен для поиска начального заполнения выбранного регистра в соответствии с частью вероятной гаммы. Блок позволяет моделировать работу регистра на некоторое число блоков назад (1 блок=8 шагов) и получать таким образом нужное начальное заполнение, которое так же представлено в этом блоке;
- Гамма. Блок предназначен для получения и отображения гаммы, которая получается используя вид регистра и его начальное заполнение;
- Открытый текст. Блок необходим для получения текстового представления открытого текста, который получен сложением шифрованного текста и гаммы по модулю 2 и последующей перекодировкой;
- Управление. Блок является вспомогательным. Он предназначен для управлением автоматическим созданием файла отчета, нахождения матрицы A (имеется кнопка «Матрицап A», вызывающая подпрограмму поиска матрицы A) и для завершения работы (в данном блоке имеется кнопка «Выход», предназначенная для завершения лабораторной работы и закрытия рабочего окна).

Требования к размещению файлов

Для запуска лабораторной работы необходимо наличие файла LSR.exe, для ее выполнения нужен файл соответсвующего варианта (всего 25 различных вариантов $\Rightarrow$ 25 файлов). Файлы вариантов должны располагаться в том же каталоге, что и LSR.exe. Заметим, что файл отчета lsr.log будет создаваться так же в том же каталоге.

Необходимые знания

Для успешного выполнения лабораторной работы требуются базовые знания в области аддитивных шифров, в частности общие понятия о принципе действия линейного сдвигового регистра, а также пользовательские навыки работы с ОС Windows. Изложенный ранее краткий теоретический материал является достаточным для выполнения лабораторной работы.

Загрузка варианта

Каждому студенту преподавателем назначается вариант, и в соответствии со своим вариантом студент выполняет лабораторную работу.

Для загрузки соответствующего варианта предназначено поле выбора и кнопка «Загрузить» в верхней части блока «Вариант»

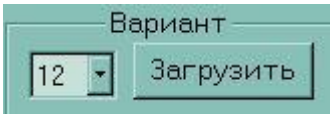


Рис 1.24.Часть блока вариант

Выполняющий работу (студент) выбирает один из предложеных 25 номеров варианта и нажимает кнопку «Загрузить»

Рис 1.25. Кнопка «Загрузить»

После нажатия кнопки в поле для чтения «Шифрограмма» появляется зашифрованный текст в битовом представлении или возникает сообщение об ошибке (см. Сообщения выдаваемые в процессе работы). Задачей выполняющего является расшифровка данного текста.

Рис 1.26. Поле для чтения «Шифрограмма»

В поле для чтения «Шифрограмма» располагается двоичное представление зашифрованного текста. Каждые восемь бит в совокупности представляют собой одну закодированную букву. Ознакомиться с кодировкой можно в Приложении 1.

Всего в поле для чтения «Шифрограмма» представлено 16 закодированных букв (128 бит), таким образом зашифрованный текст представляет собой слово или фразу из 16 символов.

Выбор вероятных составляющих

Поскольку для дальнейшего расшифрования текста (а именно отыскания начального заполнения еще неопределенного регистра) нам требуется 2*L бит гаммы (L – разрядность регистра, в работе n=7 $\Rightarrow$ требуется 14 бит), то следующим шагом в выполнении работы является определение вероятной биграммы и ее положения в зашифрованном тексте. Для этого предназначено поле выбора «Вероятная биграмма»

Рис 1.27. Поле «Вероятна биграмма»

На выбор выполняющему работу предлагается 8 биграмм (ЕН, ЕТ, НА, НИ, ПР, РА, СТ, ТО). Эти биграммы являются наиболее вероятными в русском языке, следовательно

хотябы одна из них должна содержатся в зашифрованном сообщении (см. полную таблицу вероятностей биграмм в тексте в Приложении 2).

После выбора в поле «Текст» выроятной биграммы в соседнем поле для чтения появится битовое представление этой биграммы, кроме того тоже битовое представление появится в поле ввода «Вторая стр» блока «Сложение по модулю 2».

Рис 1.28. Выбранная биграмма

На этом выбор вероятной биграммы закончен. Теперь необходимо определить ее положение в тексте. Будем последовательно перебирать все возможные положения данной биграммы при помощи двух управляющих кнопок «Предыдущая» и «Следующая» (подзаголовок Позиция в тексте).

Рис 1.29. Позиция в тексте и управляющие кнопки

При нажатии на кнопку «Следующая» или «Предыдущая» в левом поле рис 7 появится часть шифрограммы, которая соответствует позициям, номера которых появятся в правом поле для чтения. Одновременно с этим поизойдет заполнение первого поля в блоке «Сложение по модулю 2» содержимым левого поля.

После того как выбрана биграмма и ее положение (то есть заполнены два верхних поля в блоке «Сложение по модулю 2»), в поле «=» блока «Сложение по модулю 2» появится результат сложения.

На этом определение вероятного местоположения вероятной биграммы и части вероятной гаммы закончен. Таким образом мы имеем предполагаемую биграмму, ее предполагаемое местоположение и, вероятно, часть ключа. Дальнейшие шаги покажут нам правильность или ошибочность выбора предполагаемых компонентов.

Нахождение вероятной части ключа

Данный шаг необходим для ручного сложения по модулю 2 собственных компонентов, то есть на предыдущем шаге вероятная часть ключа была найдена автоматически. Таким образом данное описание можно пропустить.

Для определения вероятной части ключа мы будем использовать блок «Сложение по модулю 2» с внесенными в него на предыдущем шаге начальными данными (вероятной биграммой и соответствующей ей части зашифрованного текста).

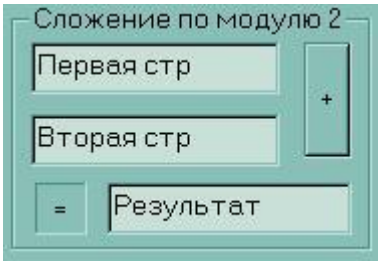


Рис 1.30. Блок «Сложение по модулю 2»

Поскольку для определения вероятной части гаммы достаточно простого сложения по модулю два вероятной биграммы и соответствующей ей части зашифрованного текста, то для получения необходимо нажать кнопку «+».

Рис 1.31. Кнопка «+» (сложить)

После чего в поле «=» появится искомая часть вероятной гаммы.

Рис 1.32. Поле «=» - результат сложения

Таким образом мы определили 16 бит ключевой последовательности, которые нужны нам для отыскания положения отводов в регистре, начального заполнения регистра и, как следствие, всей гаммы и открытого текста. Строго говоря, 2 бита из этой последовательности являются избыточными, поскольку для определения положения отводов нужно 2*7=14 бит, а для получения начального заполнения всего 7 бит, но в связи с выбранной кодировкой символов приходится учитывать и эти 2 бита.

Определение положения отводов

Одним из ключевых шагов в выполнении работы является нахождение положения отводов в регистре. В данной работе предполагается определение положения отводов при

помощи метода основанного на нахождении обратной матрицы методом Гаусса, используя подпрограмму для обращения матрицы и нахождения матрицы А.

Для определения положения отводов выполняющему необходимо вызвать подпрограмму нахождения матрицы А, нажатием кнопки «Матрица А».

Рис 1.33. Кнопка «Матрица А»

Затем в появившемся диалоговом окне необходимо заполнить поля представляющие собой поля для ввода векторов-столбцов S(1)…S(8) (см. Теоретическое введение).

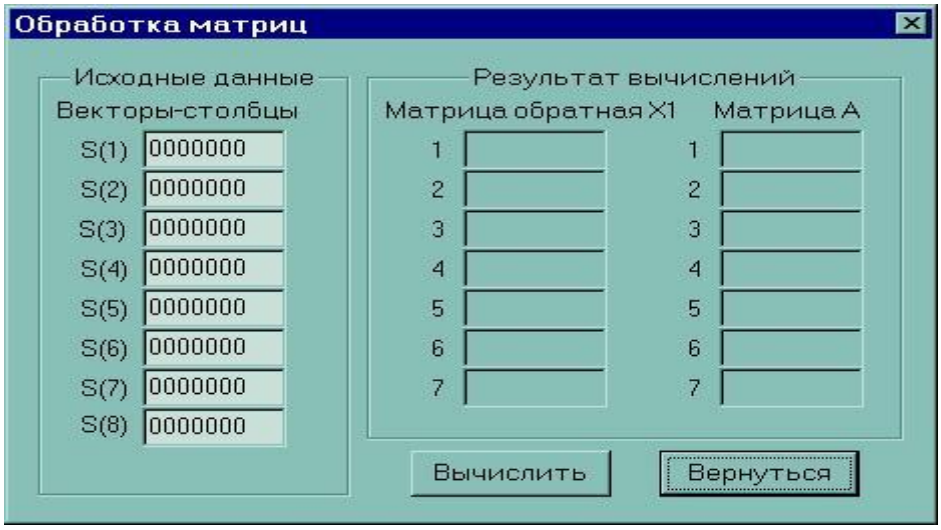


Рис 1.34. Окно подпрограммы для нахождения матрицы А

После корректного заполнения вышеуказанных полей, необходимо нажать кнопку «Вычислить» и в соответствующих полях окна подпрограммы появятся строки соттветствующие матрицам X^{-1} и А.

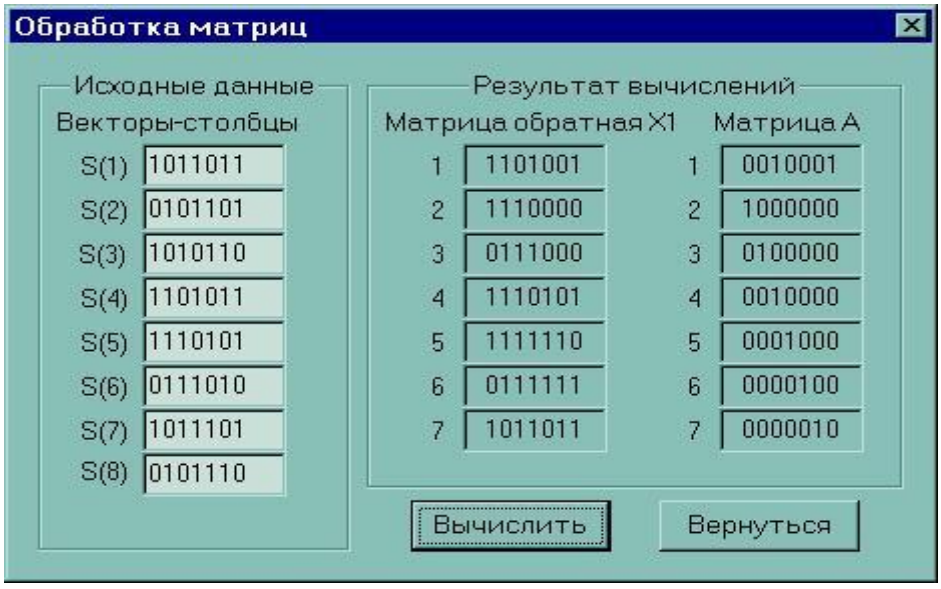


Рис 1.35. Результат работы после нажатия на кнопку «Вычислить»

Следует отметить, что матрица А должна иметь специальный вид: первая строка – определяет положение отводов, в остальных строках под главной диаганалью находятся единицы, остальные нули. Если найденная матрица отличается по виду от вышеописанной, то была допущена ошибка на ранних шагах (например выбрана ошибочная биграмма). Строка 1 подраздела «Матрица А» является определяющей, то есть именно ее вид определяет положение отводов и именно ее необходимо заносить в поле А1 блока положение отводов, после выхода из подпрограммы (нажатием кнопки «Вернуться»).

Рис 1.36. Блок положение отводов

Поскольку для определение отводов существует, по крайней мере, два способа определения положения отводов, то возможно 2 способа заполнения положения отводов. Рассмотрим эти способы.

1) Если отводы были определены при помощи нахождения обратной матрицы, то удобно ввести в поле «А1=» первую строку матрицы А, что будет являться заданием положения отводов и будет продублировано в нижней части блока.

Регистр по условию лабораторной работы является 7-разрядным, то есть первая строка матрицы А является последовательностью из 7 бит, каждый из которых говорит о наличии (если бит равен 1) или отсутствии (если бит равен 0) отвода в регистре.

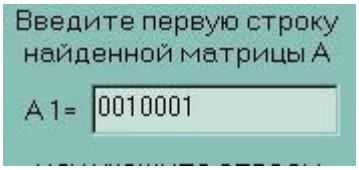


Рис 1.37. Строковое задание положения отводов

Если положение отводов были найдены другим способом, то удобно непосредственно указать отводы присутствующие в регистре, то есть активировать чек-бокс соответствующий присутствующему отводу, введенные данные продублируются в строке «А1=»

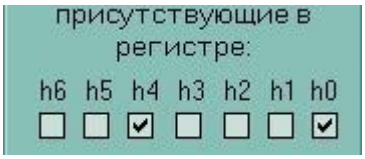


Рис 1.38. Непосредственный выбор отводов

Необходимо внимательнее подходить к проблеме поиска отводов в регистре, так как неправильное определение положения отводов влечет за собой неправильный результат.

Поиск начального заполнения

Для того, чтобы расшифровать текст необходима гамма такой же длины как и зашифрованный текст. Для получения гаммы нам нужно знать начальное заполнение регистра. Для определения начального заполнения в лабораторной работе используется блок «Начальное заполнение».

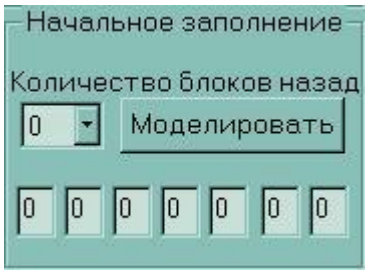


Рис 1.39. Блок «Начальное заполнение»

Поскольку для получения начального заполнения необходимо промоделировать обратную работу регистра, то существует кнопка «Моделировать», при нажатии на которую происходит обратное моделирование работы регистра на заданное количество шагов, которое задается в поле выбора «Количество блоков назад».

Рис 1.40. Кнопка «Моделировать»

Поскольку нецелесообразно моделировать обратную работу на число шагов не кратное 8 (так как 1 символ закодирован 8 битами), то число шагов заменено числом блоков. То есть 1 блок = 8 шагов, и при моделировании на 1 блок производится обратная работа на 8 шагов. Выбор количества блоков назад ограничен 14 (для обеспечния отсутствия цикличности).

Выбор количества блоков на которое производится обратное моделирование важен для правильности определения начального заполнения. Количество блоков для обратного моделирования является первой цифрой в номере позиции вероятной биграммы (см. Подзаголовок «Позиция в тексте» блока «Вариант» правое поле для чтения). То есть если позиция представлена как 3-4 (то есть вероятная биграмма находится на позиции 3 и позиции 4), то обратное моделирование должно производиться на 3 блока назад.

После нажатия на кнопку «Моделировать» автоматически производится поиск начального заполнения регистра. Для этого используются первые 7 бит строки «=» блока «Сложение по модулю 2» и регистр из блока «Положение отводов» (точнее положение его отводов). Полученный результат отображается в схематичном представлении ячеек регистра, заполненных нулями или единицами.

Рис 1.41. Схематичное представление ячеек регистра

Кроме того для удобства выполнения работы сразу после нажатия кнопки «Моделировать», если не произошло никаких ошибок заполняются поля в блоках «Гамма» и «Открытый текст». Таким образом после нажатия кнопки «Моделировать» **при правильном выборе вероятной биграммы, ее положения в тексте и правильного определения положения отводов получается открытый текст.**

Получение гаммы

Для расшифрования сообщения нам необходимо получить гамму, которая использовалась при зашифровке. Этот шаг выполняется автоматически при нажатии на кнопку «Моделировать» из блока «Начальное заполнение». Для контроля за правильностью гаммы предназначен блок «Гамма»

Рис 1.42. Блок «Гамма»

Гамма представляет собой последовательность 128 двоичных символов, которые выводятся в поле для чтения «Гамма». Данная последовательность используется для последующего сложения по модулю 2 с шифрограммой и получения открытого текста в битовом представлении.

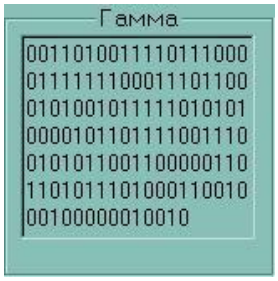


Рис 1.43. Поле для чтения «Гамма»

Получение открытого текста

Открытый текст получается автоматически при нажатии на кнопку «Моделировать» блока «Начальное заполнение», однако для контроля предусмотрены дополнительный возможности.

Открытый текст представляется в программе перекодированным из битовой последовательности в символы и для этого используется блок «ОТКРЫТЫЙ ТЕКСТ».

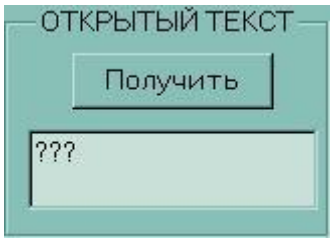


Рис 1.44. Блок «Открытый текст»

Для получения открытого текста достаточно нажать кнопку «Получить». Программа автоматически произведет сложение гаммы и зашифрованного текста, а потом перекодирует битовый текст в символьный.

Рис 1.45. Кнопка «Получить»

В результате в поле ввода появится некоторый текст, который либо представляет собой осмысленное сообщение (тогда работа успешно завершена), либо непонятный набор символов (увы, придется повторить некоторые шаги заново). Во втором случае наиболее вероятным местом ошибки является неправильно выбранное количество блоков для обратного моделирования (как следствие неправильные начальное заполнения и гамма). Если же вы уверены в своих действиях по выбору количества блоков, тогда неверно выбрана биграмма или ее положение (то есть придется вернуться к п 5.3.4), кроме того возможно неверное определение положения отводов (придется вернуться к п 5.3.5)

Если полученный открытый текст устраивает выполняющего то работа завершена.

Отчет о проделанной работе

Для контроля за выполнением работы предусмотрено специальное средство – отчет о проделанной работе. В данной лабораторной отчет представляется в форме файла отчета:

- файл отчета – необходим для предоставления проверяющему (преподавателю);

Форма отчета включаются путем выбора соответствующего элемента в блоке «Управление».

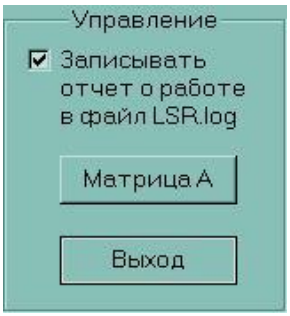


Рис 1.46. Блок «Управление»

Выключатель «Записывать отчет о работе в файл LSR.log» включает\выключает режим записи произведенных действий в файл «lsr.log».

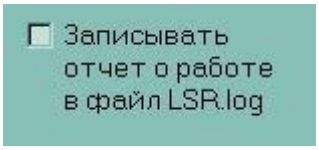


Рис 1.47. Выключатель «Записывать отчет о работе в файл LSR.log»

При включении данного выключателя создается или перезаписываетя или дозаписывается (в зависимости от ситуации) файл «lsr.log», в который записываются действия пользователя по отысканию открытого текста.

Для составления отчета надо:

a) После запуска лабораторной работы включить переключатель «Записывать отчет о работе в файл LSR.log» (включен по умолчанию). Если уже существует lsr.log, то ответить на вопрос: «Переписывать?». Если такого файла нет, то он создастся;

b) Загрузить вариант. В файле появится запись «Начало LOG*********»;

c) Выполнить действия по поиску открытого текста;

d) Найти открытый текст.

e) Выйти из программы при помощи кнопки «Выход». В файле появится запись «Конец LOG*********».

Рис 1.48. Кнопка «Выход»

В файле отчета будут задокументированны основные действия по поиску открытого текста. Отчет предоставляется в распечатанном виде от фразы «Начало LOG**********» до фразы «Конец LOG**********».

Лабораторная работа 3. Исследование Российского алгоритма симметричного шифрования ГОСТ 28147-89

Цель работы Изучить криптографический стандарт шифрования ГОСТ 28147-89 и его особенности, познакомиться с различными режимами блочного шифрования.

ГОСТ 28147-89 — советский и российский стандарт симметричного шифрования, введённый в 1990 году, также является стандартом СНГ. Полное название — «ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования». С момента опубликования ГОСТа на нём стоял ограничительный гриф «Для служебного пользования», и формально шифр был объявлен «полностью открытым» только в мае 1994 года. К сожалению, история создания шифра и критерии разработчиков до сих пор не опубликованы.

Алгоритм криптографического преобразования предназначен для аппаратной или программной реализации, удовлетворяет криптографическим требованиям и по своим возможностям не накладывает ограничений на степень секретности защищаемой информации.

Стандарт обязателен для организаций, предприятий и учреждений, применяющих криптографическую защиту данных, хранимых и передаваемых в сетях, в отдельных вычислительных комплексах или на персональных компьютерах.

То, что в его названии вместо термина «шифрование» фигурирует более общее понятие «криптографическое преобразование», вовсе не случайно. Помимо нескольких тесно связанных между собой процедур шифрования, в документе описан один построенный на общих принципах с ними алгоритм выработки имитовставки. Последняя является не чем иным, как криптографической контрольной комбинацией, то есть кодом, вырабатываемым из исходных данных с использованием секретного ключа с целью имитозащиты, или защиты данных от внесения в них несанкционированных изменений.

Математические операции

Сложение по модулю 2

Операция поразрядного XOR (обозначается как ⊕) — булева функция и логическая операция. Результат выполнения операции является истинным только при условии, если является истинным в точности один из аргументов. Пример выполнения операции сложения:

$(x^6 + x^4 + x^2 + x + 1) \oplus (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (в виде многочленов)

{01010111} ⊕ {10000011} = {11010100} (двоичное представление)

Правила суммирования по модулю 2^{32} и по модулю (2^{32}-1)

1. Два целых числа a, b, где $0 \le a$, $b \le <, 2^{32}$-1, представленные в двоичном виде

$$a = (a_{32}, a_{31}, ..., a_2, a_1),\ b = (b_{32}, и_{31}, ..., b_2, b_1),$$

т.е. $a = a_{32} \cdot 2^{31} + a_{31} \cdot 2^{30} + ... + a_2 \cdot 2 + a_1$,

$$b = b_{32} \cdot 2^{31} + b_{31} \cdot 2^{30} + ... + b_2 \cdot 2 + b_1,$$

суммируются по модулю 2^{32} (операция ⊞) по следующему правилу:

$$a \boxplus b = a + b, \text{ если } a + b < 2^{32},$$

$$a \boxplus b = a + b - 2^{32}, \text{ если } a + b \ge 2^{32},$$

где операция +(-) есть арифметическая сумма (разность) двух целых чисел.

2. Два целых числа a, b, где $0 \le a$, $b \le 2^{32}$ -1, представленные в двоичном виде

$$a = (a_{32}, a_{31}, \dots, a_2, a_1),$$

$$b = (b_{32}, b_{31}, \dots, b_2, b_1),$$

суммируются по модулю (2^{32}-1) (операция ⊞') по следующему правилу:

$$a \boxplus' b = a + b, \text{ если } a + b < 2^{32},$$

$$a \boxplus' b = a + b - 2^{32} + 1, \text{ если } a + b \ge 2^{32}.$$

Структура алгоритма

Симметричное шифрование — способ шифрования, в котором для зашифровки и расшифровывания применяется один и тот же криптографический ключ. До изобретения схемы асимметричного шифрования единственным существовавшим способом являлось симметричное шифрование. Ключ алгоритма должен сохраняться в секрете обеими сторонами. Алгоритм шифрования выбирается сторонами до начала обмена сообщениями. К симметричному шифрованию предъявляются следующие требования:

- Отсутствие линейности (то есть условия f(a) xor f(b) == f(a xor b)), в противном случае облегчается применение дифференциального криптоанализа к шифру. (Функция xor – «сложение по модулю 2», «исключающее или» – результат выполнения операции является истинным только при условии, если является истинным в точности один из аргументов).
- Полная утрата всех статистических закономерностей исходного сообщения.

Алгоритм ГОСТ 28147-89 является *блочным шифром* – разновидность симметричного шифра. Особенностью блочного шифра является обработка блока нескольких байт за одну итерацию (в нашем случае 8). Как и большинство современных блочных шифров, ГОСТ основан на *сети Фейстеля* (рисунок 7.1). Сеть представляет собой определённую

7многократно повторяющуюся (итерированную) структуру, называющуюся ячейкой Фейстеля. При переходе от одной ячейки к другой меняется ключ, причём выбор ключа зависит от конкретного алгоритма. Операции шифрования и расшифрования на каждом этапе очень просты, и при определённой доработке совпадают, требуя только обратного порядка используемых ключей. Шифрование при помощи данной конструкции легко реализуется как на программном уровне, так и на аппаратном, что обеспечивает широкие возможности применения.

Если внимательно изучить оригинал ГОСТ 28147–89, можно заметить, что в нем содержится описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня, называемые в тексте ГОСТа циклами. Эти фундаментальные алгоритмы будут называться «базовые циклы», чтобы отличать их от всех прочих циклов. Они имеют следующие названия и обозначения: (последние приведены в скобках)

- цикл зашифрования (32-З);
- цикл расшифрования (32-Р);
- цикл выработки имитовставки (16-З).

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной единственной процедуры, называемой «основным шагом криптопреобразования».

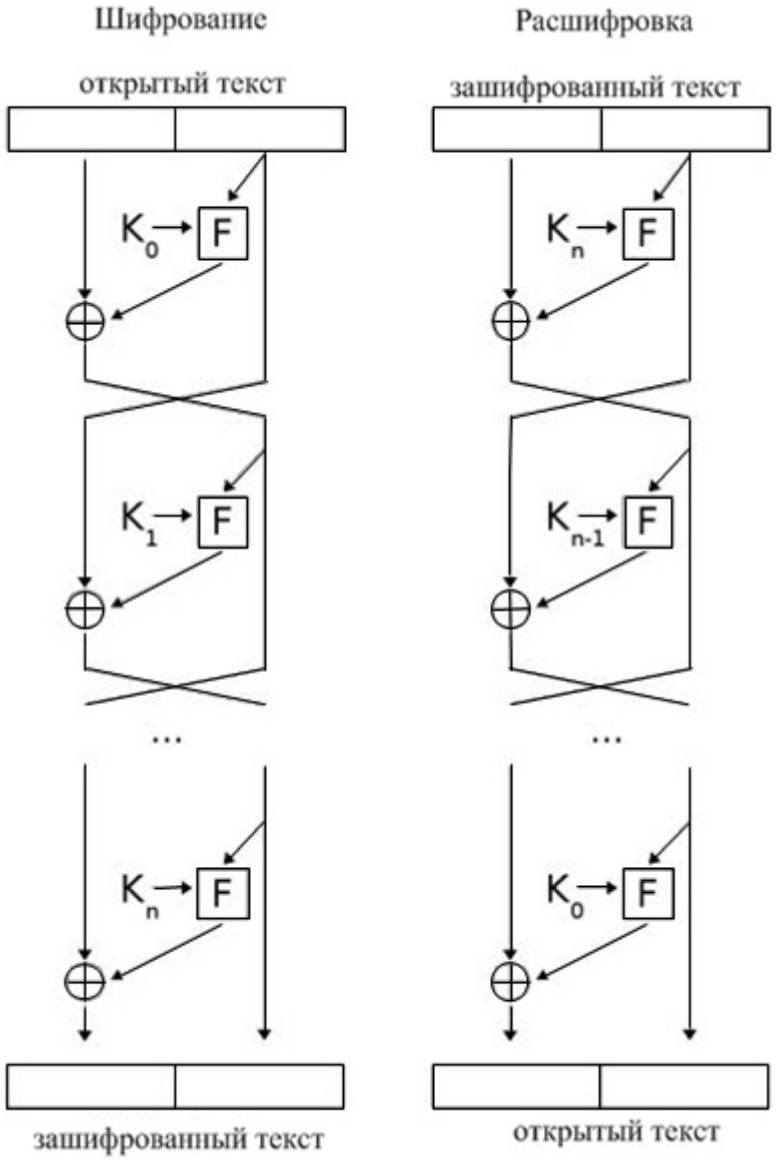


Рис. 3.1. Сеть Фейстеля

- *Ключ* является массивом из восьми 32-битовых элементов кода В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака. Таким образом, размер ключа составляет 256 бит (32 байта). Ключ должен являться массивом статистически независимых битов, принимающих с равной вероятностью значения 0 и 1. При этом некоторые конкретные значения ключа могут оказаться «слабыми», то есть шифр может не обеспечивать заданный уровень стойкости в случае их использования. Однако, предположительно, доля таких значений в общей массе всех возможных ключей ничтожно мала. Поэтому ключи, выработанные с помощью некоторого датчика истинно случайных чисел, будут качественными с вероятностью, отличающейся от единицы на ничтожно малую величину.

- *Таблица замен* является вектором, содержащим восемь *узлов замены*. Каждый узел замены, в свою очередь, является вектором, содержащим шестнадцать 4-битовых

элементов замены, которые можно представить в виде целых чисел от 0 до 15, все элементы одного узла замены обязаны быть различными. Таким образом, таблица замен может быть представлена в виде матрицы размера 8x16 или 16x8, содержащей 4-битовые заменяющ Таким образом, общий объем таблицы замен равен 512 бит (64 байта).

Общее количество узлов замены (S-блоков) ГОСТа — восемь. Каждый S-блок представляет собой перестановку чисел от 0 до 15. Первая 4-битная подпоследовательность попадает на вход первого S-блока, вторая — на вход второго и т. д.

Если S-блок выглядит так: 1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12

и на входе S-блока 0, то на выходе будет 1, если 4, то на выходе будет 5, если на входе 12, то на выходе 6 и т. д. (для таблицы 2.24)

Таблица 3.1. S-блоки, приведенные в ГОСТ Р 34.11-94 для целей тестирования

Номер S-блока	Значение															
1	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
2	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
3	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
4	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
5	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
6	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
7	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
8	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Основной шаг криптопреобразования

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена на рисунке 2.2. Рассмотрим подробнее этапы основного шага криптопреобразования:

Шаг 0 Определяет исходные данные для основного шага криптопреобразования.

N – преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать $N=(N_1,N_2)$.

X – 32-битовый элемент ключа;

Шаг 1 Сложение с ключом.

Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий шаг;

Шаг 2 Поблочная замена.

32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода:

$$S=(S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$$

причем S_0 содержит 4 самых младших, а S_7 – 4 самых старших бита S. Далее значение каждого из восьми блоков заменяется новым, которое выбирается по таблице замен следующим образом: значение блока S_i меняется на S_i-тый по порядку элемент (нумерация с нуля) i-того узла замены (т.е. i-той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Отсюда становится понятным размер таблицы замен: число строк в ней равно числу 4-битовых элементов в 32-битовом блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битового блока данных, равному, как известно 2^4 (т. е. шестнадцати).

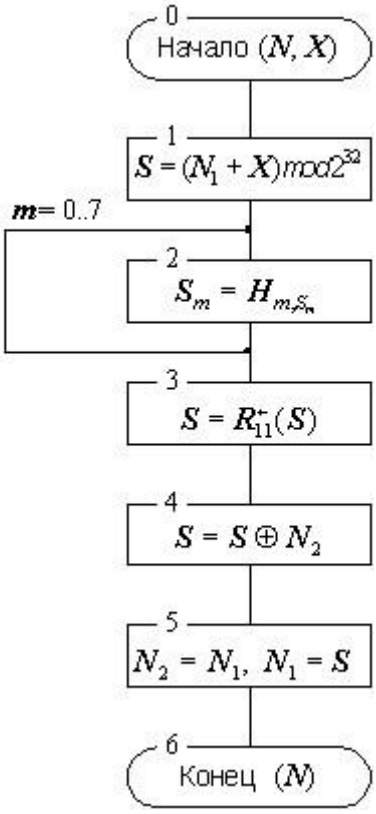


Рис. 3.2. Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89

Шаг 3 Циклический сдвиг на 11 бит влево.

Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом «$R_{11}^{\leftarrow}$»

обозначена функция циклического сдвига своего аргумента на 11 бит влево, т.е. в сторону старших разрядов.

Шаг 4 Побитовое сложение

Значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Шаг 5 Сдвиг по цепочке

Младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Шаг 6 Завершение криптопреобразования.

Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

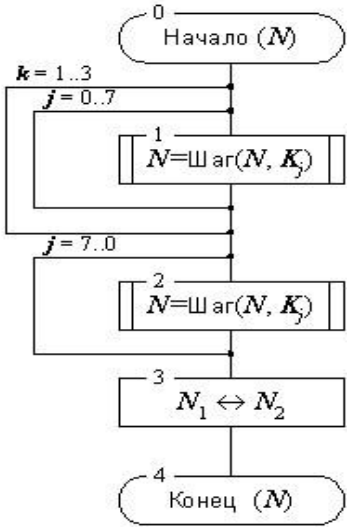


Рис. 3.3. Схема цикла зашифрования 32-З

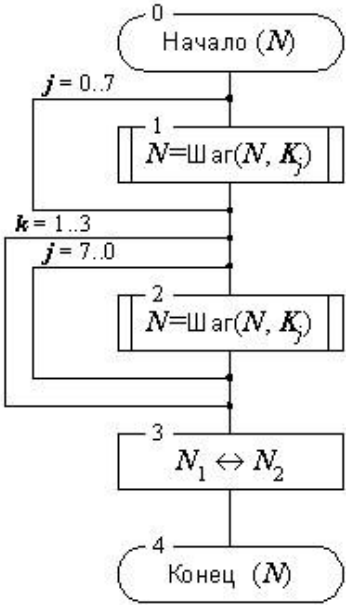


Рис.3.4. Схема цикла расшифрования 32-Р

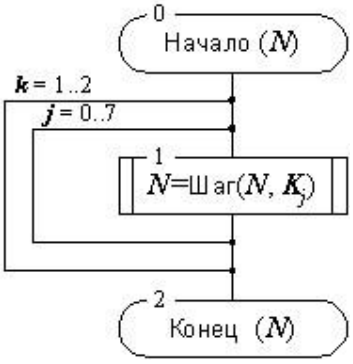


Рис. 3.5. Схема цикла выработки имитовставки 16-З.

Порядок использования ключевых элементов:

- Цикл зашифрования 32-З:

$K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_7,K_6,$
K_5,K_4,K_3,K_2,K_1,K_0

- Цикл расшифрования 32-Р:

$K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_7,K_6,K_5,K_4,K_3,K_2,K_1,K_0,K_7,K_6,K_5,K_4,K_3,K_2,K_1,K_0,K_7,K_6,K_5,K_4,K_3,K_2,K_1,K_0$

- Цикл выработки имитовставки 16-З:

$K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7$

Режимы шифрования

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,
- дополнительный режим выработки имитовставки

Простая замена

Зашифрование в данном режиме заключается в применении цикла 32-З к блокам открытых данных, расшифрование – цикла 32-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены на рисунках 2.43 и 2.44 соответственно. Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам:

$$|T_o|=|T_ш|=64\cdot n$$

где n – целое положительное число

после выполнения операции размер полученного массива данных не изменяется.

Режим шифрования простой заменой имеет следующие особенности:

- Так как блоки данных шифруются независимо друг от друга и от их позиции в массиве данных, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки шифртекста и наоборот. Отмеченное свойство позволит криптоаналитику сделать заключение о тождественности блоков исходных данных, если в массиве зашифрованных данных ему встретились идентичные блоки, что является недопустимым для серьезного шифра.
- Если длина шифруемого массива данных не кратна 8 байтам (64 битам), возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит. Эта задача не так проста, как кажется на первый взгляд. Очевидные решения вроде «дополнить неполный блок нулевыми битами» или, более общее, «дополнить неполный блок фиксированной комбинацией нулевых и единичных битов» могут при определенных условиях дать в руки криптоаналитика возможность методами перебора определить содержимое этого самого неполного блока, и этот факт означает снижение стойкости шифра. Кроме того, длина шифртекста при этом изменится, увеличившись до ближайшего целого, кратного 64 битам, что часто бывает нежелательным.

На первый взгляд, перечисленные выше особенности делают практически невозможным использование режима простой замены, ведь он может применяться только для шифрования массивов данных с размером кратным 64 битам, не содержащим повторяющихся 64-битовых блоков. Кажется, что для любых реальных данных гарантировать выполнение указанных условий невозможно. Это почти так, но есть одно очень важное исключение: вспомните, что размер ключа составляет 32 байта, а размер таблицы замен – 64 байта. Кроме того, наличие повторяющихся 8-байтовых блоков в ключе или таблице замен будет говорить об их весьма плохом качестве, поэтому в реальных ключевых элементах такого повторения быть не может. Таким образом, мы выяснили, что режим простой замены вполне подходит для шифрования ключевой информации, тем более, что прочие режимы для этой цели менее удобны, поскольку требуют наличия дополнительного синхронизирующего элемента данных – синхропосылки (см. следующий подпункт). И поэтому ГОСТ предписывает использовать режим простой замены исключительно для шифрования ключевых данных.

Гаммирование

Как же можно избавиться от недостатков режима простой замены? Для этого необходимо сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифртекста от его номера, иными словами, рандомизировать процесс

шифрования. В ГОСТе это достигается двумя различными способами в двух режимах шифрования, предусматривающих гаммирование.

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Для наложения гаммы при зашифровании и ее снятия при расшифровании должны использоваться взаимно обратные бинарные операции. В ГОСТе для этой цели используется операция побитового сложения по модулю 2, поскольку она является обратной самой себе и, к тому же, наиболее просто реализуется аппаратно. Гаммирование решает обе упомянутые проблемы: во-первых, все элементы гаммы различны для реальных шифруемых массивов и, следовательно, результат зашифрования даже двух одинаковых блоков в одном массиве данных будет различным. Во-вторых, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока.

Гамма для этого режима получается следующим образом: с помощью некоторого алгоритмического рекуррентного генератора последовательности чисел (РГПЧ) вырабатываются 64-битовые блоки данных, которые далее подвергаются преобразованию по циклу 32-З, то есть зашифрованию в режиме простой замены, в результате получаются блоки гаммы. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции побитового исключающего или, алгоритмы зашифрования и расшифрования в режиме гаммирования идентичны.

РГПЧ, используемый для выработки гаммы, является рекуррентной функцией:

$$\Omega_{i+1} = f(\Omega_i),$$

где Ω_i - элементы рекуррентной последовательности,

f – функция преобразования

Элемент данных «Ω_0» является параметром алгоритма для режимов гаммирования, на схемах он обозначен как S, и называется в криптографии *синхропосылкой,* а в ГОСТе – *начальным заполнением* одного из регистров шифрователя.

Синхропосылка (вектор инициализации) – случайное число, которое регулярно обновляется, передается по каналу управления и используется для инициализации алгоритма шифрования.

Из соображений увеличения стойкости алгоритма, разработчики ГОСТа решили использовать для инициализации РГПЧ не непосредственно синхропосылку, а результат ее преобразования по циклу 32-З.

Последовательность элементов, вырабатываемых РГПЧ, целиком зависит от его начального заполнения, то есть элементы этой последовательности являются функцией своего номера и начального заполнения РГПЧ:

$$\Omega_i = f_i(\Omega_0),$$

где $f_i(X)=f(f_{i-1}(X))$,

$f_0(X)=X$

С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа:

$$\Gamma_i = U_{32\text{-}3}(\Omega_i) = U_{32\text{-}3}(f_i(\Omega_0)) = U_{32\text{-}3}(f_i(U_{32\text{-}3}(S))) = \varphi_i(S,K),$$

где Γ_i – i-тый элемент гаммы,

K – ключ

Естественно, для обратимости процедуры шифрования в процессах за- и расшифрования должна использоваться одна и та же синхропосылка. Из требования уникальности гаммы, невыполнение которого приводит к катастрофическому снижению стойкости шифра, следует, что для шифрования двух различных массивов данных на одном ключе необходимо обеспечить использование различных синхропосылок. Это приводит к необходимости хранить или передавать синхропосылку по каналам связи вместе с зашифрованными данными, хотя в отдельных особых случаях она может быть предопределена или вычисляться особым образом, если исключается шифрование двух массивов на одном ключе.

Подробнее рассмотрим РГПЧ, используемый в ГОСТе для генерации элементов гаммы. Прежде всего, надо отметить, что к нему не предъявляются требования обеспечения каких-либо статистических характеристик вырабатываемой последовательности чисел. РГПЧ спроектирован разработчиками ГОСТа исходя из необходимости выполнения следующих условий:

- период повторения последовательности чисел, вырабатываемой РГПЧ, не должен сильно (в процентном отношении) отличаться от максимально возможного при заданном размере блока значения 2^{64};
- соседние значения, вырабатываемые РГПЧ, должны отличаться друг от друга в каждом байте, иначе задача криптоаналитика будет упрощена;

- РГПЧ должен быть достаточно просто реализуем как аппаратно, так и программно на наиболее распространенных типах процессоров, большинство из которых, как известно, имеют разрядность 32 бита

Исходя из перечисленных принципов, создатели ГОСТа спроектировали весьма удачный РГПЧ, имеющий следующие характеристики:

- в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга, фактически, существуют два независимых РГПЧ для старшей и младшей частей блока:

$$\Omega_i = (\Omega_i^0, \Omega_i^1),$$

$$|\Omega_i^0| = |\Omega_i^1| = 32,$$

$$\Omega_{i+1}^0 = \hat{f}(\Omega_i^0),$$

$$\Omega_{i+1}^1 = \tilde{f}(\Omega_i^1)$$

- рекуррентные соотношения для старшей и младшей частей следующие (нижний индекс в записи числа означает его систему счисления, таким образом, константы, используемые на данном шаге, записаны в 16-ричной системе счисления):

$$\Omega_{i+1}^0 = (\Omega_i^0 + C_0) mod 2^{32}$$

где $C_0=1010101_{16}$

$$\Omega_{i+1}^1 = (\Omega_i^1 + C_1 - 1) mod (2^{32} - 1) + 1$$

где $C_1=1010104_{16}$

Второе выражение нуждается в комментариях, так как в тексте ГОСТа приведено нечто другое:

$$\Omega_{i+1}^1 = (\Omega_i^1 + C_1) mod (2^{32} - 1)$$

тем же значением константы C_1. Но далее в тексте стандарта дается комментарий, что, оказывается, под операцией взятия остатка по модулю $(2^{32}–1)$ там понимается не то же самое, что и в математике. Отличие заключается в том, что согласно ГОСТу:

$$(2^{32}–1) \bmod (2^{32}–1) = (2^{32}–1), \text{а не } 0$$

Период повторения последовательности для младшей части составляет 2^{32}, для старшей части $(2^{32}–1)$, для всей последовательности период составляет $2^{32}*(2^{32}–1)$. Первая формула из двух реализуется за одну команду, вторая , несмотря на ее кажущуюся громоздкость, за две команды на всех современных 32-разрядных процессорах – первой командой идет обычное сложение по модулю 2^{32} с запоминанием бита переноса, а вторая команда прибавляет бит переноса к полученному значению.

Рассмотрим шаги алгоритма гаммирования, представленного на рисунке 3.6.

Шаг 0 Определяет исходные данные для основного шага криптопреобразования.

$T_{о(ш)}$ – массив открытых (зашифрованных) данных произвольного размера, подвергаемый процедуре зашифрования (расшифрования), по ходу процедуры массив подвергается преобразованию порциями по 64 бита;

S – синхропосылка , 64-битовый элемент данных, необходимый для инициализации генератора гаммы;

Шаг 1 Начальное преобразование синхропосылки.

Выполняется для ее «рандомизации», то есть для устранения статистических закономерностей, присутствующих в ней, результат используется как начальное заполнение РГПЧ;

Шаг 2 Один шаг работы РГПЧ, реализующий его рекуррентный алгоритм.

В ходе данного шага старшая (S_1) и младшая (S_0) части последовательности данных вырабатываются независимо друг от друга;

Шаг 3 Гаммирование.

Очередной 64-битовый элемент, выработанный РГПЧ, подвергается процедуре зашифрования по циклу 32–З, результат используется как элемент гаммы для зашифрования (расшифрования) очередного блока открытых (зашифрованных) данных того же размера;

Шаг 4 Завершение гаммирования.

Результат работы алгоритма – зашифрованный (расшифрованный) массив данных.

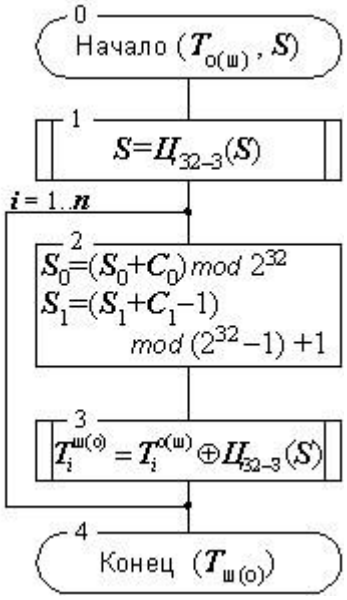


Рис. 3.6. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования

Особенности гаммирования как режима шифрования:

1. Одинаковые блоки в открытом массиве данных дадут при зашифровании различные блоки шифртекста, что позволит скрыть факт их идентичности.

2. Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо как шифрование битов этого неполного блока, для чего используется соответствующие биты блока гаммы. Так, для зашифрования неполного блока в 1 бит согласно стандарту следует использовать самый младший бит из блока гаммы.

3. Синхропосылка, использованная при зашифровании, каким-то образом должна быть передана для использования при расшифровании. Это может быть достигнуто следующими путями:

- хранить или передавать синхропосылку вместе с зашифрованным массивом данных, что приведет к увеличению размера массива данных при зашифровании на размер синхропосылки, то есть на 8 байт;
- использовать предопределенное значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует

4. Биты массива данных шифруются независимо друг от друга. Таким образом, каждый бит шифртекста зависит от соответствующего бита открытого текста и, естественно, порядкового номера бита в массиве. Из этого вытекает, что изменение бита шифртекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный. Данное свойство дает злоумышленнику возможность, воздействуя на биты шифртекста, вносить предсказуемые и даже целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифрования, не обладая при этом секретным ключом. Это иллюстрирует хорошо известный в криптологии факт, что *секретность* и *аутентичность* различные свойства криптографических систем.

Гаммирование с обратной связью

Данный режим очень похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-З предыдущего блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки (рисунок 2.7). Этим достигается зацепление блоков – каждый блок шифртекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста, это видно и в уравнении режима шифрования (расшифрования) гаммирования с обратной связью (формула 2.25). Поэтому данный режим иногда называется *гаммированием с зацеплением блоков*. На стойкость шифра факт зацепления блоков не оказывает никакого влияния.

Шифрование в режиме гаммирования с обратной связью обладает теми же особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифртекста на соответствующий открытый текст (свойство 4).

$$T_i^{o} = T_i^{ш} \oplus Ц_{32-3}(T_{i-1}^{ш})$$

Если в режиме обычного гаммирования изменения в определенных битах шифртекста влияют только на соответствующие биты открытого текста, то в режиме гаммирования с обратной связью картина несколько сложнее. Как видно из соответствующего уравнения, при расшифровании блока данных в режиме гаммирования с обратной связью, блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифрованный блок, то после расшифрования искаженными окажутся два блока открытых данных – соответствующий и следующий за ним, причем искажения в первом случае будут носить тот же характер, что и в режиме гаммирования, а во втором случае – как в режиме простой замены. Другими словами, в соответствующем блоке открытых данных искаженными окажутся те же самые биты, что и в блоке шифрованных данных, а в следующем блоке открытых данных все биты независимо друг от друга с вероятностью 1/2 изменят свои значения.

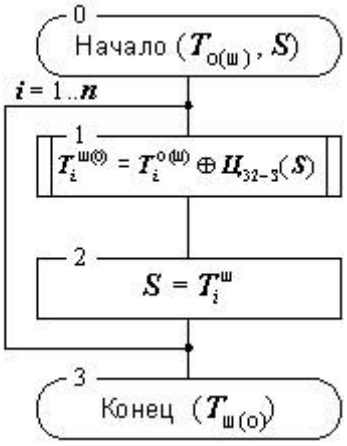


Рис. 3.7. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования с обратной связью.

Выработка имитовставки к массиву данных

Для решения задачи обнаружения искажений в зашифрованном массиве данных с заданной вероятностью в ГОСТе предусмотрен дополнительный режим криптографического преобразования – выработка имитовставки (рисунок 3.8).

Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации. Для потенциального злоумышленника две следующие задачи практически неразрешимы, если он не владеет ключом:

- вычисление имитовставки для заданного открытого массива информации;

- подбор открытых данных под заданную имитовставку

В качестве имитовставки берется часть блока, полученного на выходе, обычно – 32 его младших бита. При выборе размера имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных равна величине $2^{-|I|}$ на одну попытку подбора, если в распоряжении злоумышленника нет более эффективного метода подбора, чем простое угадывание. При использовании имитовставки размером 32 бита эта вероятность равна $2^{-32} \approx 0.23 \cdot 10^{-9}$.

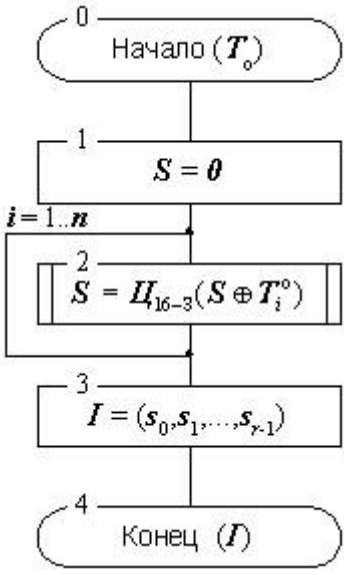


Рис. 3.8. Алгоритм выработки имитовставки для массива данных

Криптоанализ алгоритма

В 1994 г. описание алгоритма ГОСТ 28147-89 было переведено на английский язык и опубликовано в статье Волонгонгского университета (Австралия) «Советский алгоритм шифрования». Именно после этого стали появляться результаты его анализа, выполненного зарубежными специалистами; однако, в течение значительного времени не было найдено каких-либо атак, приближающихся к практически осуществимым.

Анализ таблиц замен

Поскольку таблицы замен в стандарте не приведены, в ряде работ высказывается предположение, что «компетентная организация» может выдать как «хорошие», так и «плохие» таблицы замен. Ясно, что криптостойкость алгоритма во многом зависит от свойств используемых таблиц замен, соответственно, существуют слабые таблицы замен, применение которых может упростить вскрытие алгоритма. Тем не менее, возможность использования различных таблиц замен кажется весьма достойной идеей, в пользу которой можно привести два следующих факта из истории стандарта шифрования США DES:

- Атаки с помощью как линейного, так и дифференциального криптоанализа алгоритма DES используют конкретные особенности таблиц замен. При использовании других таблиц криптоанализ придется начинать сначала
- Были попытки усилить DES против линейного и дифференциального криптоанализа путем использования более стойких таблиц замен. Такие таблицы, действительно более стойкие, были предложены, например, в алгоритме s^5DES. Но, увы, заменить DES на s^5DES было невозможно, поскольку таблицы замен жестко определены в стандарте, соответственно, реализации алгоритма наверняка не поддерживают возможность смены таблиц на другие

Однако, в работе доказано, что секретные таблицы замен могут быть вычислены с помощью следующей атаки, которая может быть применена практически:

1. Устанавливается нулевой ключ и выполняется поиск «нулевого вектора». Этот этап занимает порядка 2^{32} операций шифрования.
2. С помощью нулевого вектора вычисляются значения таблиц замен, что занимает не более 2^{11} операций.

Модификации алгоритма и их анализ

Алгоритмы:

- GOST-H, в котором, относительно оригинального алгоритма, изменен порядок использования подключей, а именно, в раундах с 25-го по 32-й подключи используются в прямом порядке, т.е. точно так же, как и в предыдущих раундах алгоритма;
- 20-раундовый алгоритм GOSTÅ, в раунде которого для наложения ключа используется операция XOR вместо сложения по модулю 2^{32}.

По результатам анализа сделан вывод о том, что GOST-H и GOSTÅ слабее исходного алгоритма ГОСТ 28147-89, поскольку оба имеют классы слабых ключей.

Анализ полнораундового алгоритма

Существуют атаки и на полнораундовый ГОСТ 28147-89 без каких-либо модификаций. Одна из первых открытых работ, в которых был проведен анализ алгоритма посвящена атакам, использующим слабости процедур расширения ключа ряда известных алгоритмов шифрования. В частности, полнораундовый алгоритм ГОСТ 28147-89 может быть вскрыт с помощью дифференциального криптоанализа на связанных ключах, но только в случае использования слабых таблиц замен. 24-раундовый вариант алгоритма (в котором отсутствуют первые 8 раундов) вскрывается аналогичным образом при любых таблицах замен, однако, сильные таблицы замен делают такую атаку абсолютно непрактичной.

Отечественные ученые А.Г. Ростовцев и Е.Б. Маховенко в 2001 г. предложили принципиально новый метод криптоанализа (по мнению авторов, существенно более

эффективный, чем линейный и дифференциальный криптоанализ) путем формирования целевой функции от известного открытого текста, соответствующего ему шифртекста и искомого значения ключа и нахождения ее экстремума, соответствующего истинному значению ключа. Они же нашли большой класс слабых ключей алгоритма ГОСТ 28147-89, которые позволяют вскрыть алгоритм с помощью всего 4-х выбранных открытых текстов и соответствующих им шифртекстов с достаточно низкой сложностью.

В 2004 г. группа специалистов из Кореи предложила атаку, с помощью которой, используя дифференциальный криптоанализ на связанных ключах, можно получить с вероятностью 91,7% 12 бит секретного ключа. Для атаки требуется 2^{35} выбранных открытых текстов и 2^{36} операций шифрования. Как видно, данная атака, практически, бесполезна для реального вскрытия алгоритма.

Достоинства ГОСТа

- бесперспективность силовой атаки, т.е. полным перебором (XSL-атаки в учёт не берутся, т.к. их эффективность на данный момент полностью не доказана);
- эффективность реализации и соответственно высокое быстродействие на современных компьютерах.
- наличие защиты от навязывания ложных данных (выработка имитовставки) и одинаковый цикл шифрования во всех четырех алгоритмах ГОСТа

Недостатки ГОСТа

Основные проблемы ГОСТа связаны с неполнотой стандарта в части генерации ключей и таблиц замен. Тривиально доказывается, что у ГОСТа существуют «слабые» ключи и таблицы замен, но в стандарте не описываются критерии выбора и отсева «слабых». Также стандарт не специфицирует алгоритм генерации таблицы замен (S-блоков). С одной стороны, это может являться дополнительной секретной информацией (помимо ключа), а с другой, поднимает ряд проблем:

- нельзя определить криптостойкость алгоритма, не зная заранее таблицы замен;
- реализации алгоритма от различных производителей могут использовать разные таблицы замен и могут быть несовместимы между собой;
- возможность преднамеренного предоставления слабых таблиц замен лицензирующими органами;
- потенциальная возможность (отсутствие запрета в стандарте) использования таблиц замены, в которых узлы не являются перестановками, что может привести к чрезвычайному снижению стойкости шифра.

Аппаратные шифраторы серии Криптон

Устройства криптографической защиты данных серии КРИПТОН — это аппаратные шифраторы для IBM PC-совместимых компьютеров. Устройства применяются в составе средств и систем криптографической защиты данных для обеспечения информационной безопасности (в том числе защиты с высоким уровнем секретности) в государственных и коммерческих структурах.

Устройства КРИПТОН гарантируют защиту информации, обрабатываемой на персональном компьютере и/или передаваемой по открытым каналам связи.

Устройства КРИПТОН выполнены в виде плат расширения ISA и PCI персонального компьютера с процессором i386 и выше.

Устройства КРИПТОН разработаны, производятся и реализуются фирмой АНКАД. Они построены на разработанных фирмой АНКАД специализированных 32-разрядных шифрпроцессорах серии БЛЮМИНГ.

За 10 лет работы Фирма АНКАД поставила более 15 тысяч устройств КРИПТОН заказчикам в Центральном Банке, Федеральном агентстве правительственной связи и информации при Президенте РФ, министерствах обороны и внутренних дел, Министерстве по налогам и сборам, Федеральном казначействе, коммерческих банках, финансовых и страховых компаниях, многим корпоративным клиентам.

Сеть кооперационного производства устройств КРИПТОН охватывает наиболее известные предприятия российской электроники (ОАО “Ангстрем” и др.).

Устройства серии КРИПТОН имеют сертификаты соответствия требованиям ФСБ и ФСТЭК (в том числе в составе абонентских пунктов и автоматизированных рабочих мест для защиты информации, содержащей сведения, составляющие государственную тайну).

Таблица 3.2. Основные технические данные и характеристики серии КРИПТОН

Алгоритм шифрования	ГОСТ 28147-89
Размерность секретного ключа шифрования, бит	256 (количество возможных комбинаций ключей — 10^{77})
Размерность открытого ключа, бит	512 или 1024
Количество уровней ключевой системы	3 (главный ключ — пользовательский/сетевой ключ — сеансовый ключ)
Датчик случайных чисел	аппаратный (аттестован экспертной организацией)
Отклонение распределения	не более 0,0005

значения случайных чисел от равновероятного распределения	
Поддерживаемые операционные системы	MS-DOS, Windows 95(98)/ME/NT 4.0/2000/XP/2003 UNIX (Solaris/Intel) (возможно создание оригинальных программных драйверов для работы устройств)
Скорость передачи данных по сети	10/100 Мбит/с
Скорость шифрования	до 9 Мбайт/с – сетевые шифраторы до 240 Мбит/с – шифраторы жесткого диска
Класс защиты по классификации ФСБ	до КС1
Алгоритм ЭЦП	ГОСТ Р 34.10-94 (с длинной секретного ключа 256 бит)

Описание лабораторной установки и методики измерений

Интерфейс учебно-программного комплекса

Лабораторный комплекс был разработан на языке высокого уровня Visual Basic 6.

Комплекс позволяет работать только с введенным текстом (обработка ведется в двоичном коде, преобразование к которому производится посредством ASCII кодов).

Для удобства работы комплекс оснащен всплывающими подсказками на все входные и выходные параметры, рабочие поля, активные элементы и элементы управления.

Оптимизация кода программы не производилась, поэтому при работе с большими об’емами текста возможны некорректное отображение резальтата или зависание программы.

Основные элементы комплекса:

1. Поле для ввода ключа – максимальная длина ключа 32 символа, при его отсутствии будет использован нулевой вектор

2. Выбор режима шифрования:

- При выборе простой замены есть возможность проверить кратность текста 8 байтовым блокам;
- При выборе режимов гаммирование есть возможность ввести вектор инициализации (синхропосылку), при ее отсутствии будет использован нулевой вектор

3. Поле для ввода текста для зашифрования
4. Возможность очистить поле шифрования
5. Возможность узнать об’ем текста в поле шифрования
6. Кнопка для зашифрования текста
7. Время, затраченное на зашифрования – напротив поля текста
8. Поле зашифрованного текста
9. Возможность очистить поле зашифрованного текста
10. Возможность узнать об’ем зашифрованного текста
11. Кнопка для расшифрования текста
12. Время, затраченное на расшифровку – напротив поля шифротекста
13. Системное время вашего компьютера
14. Возможность использовать материалы в электронном виде (рисунок 3.9)

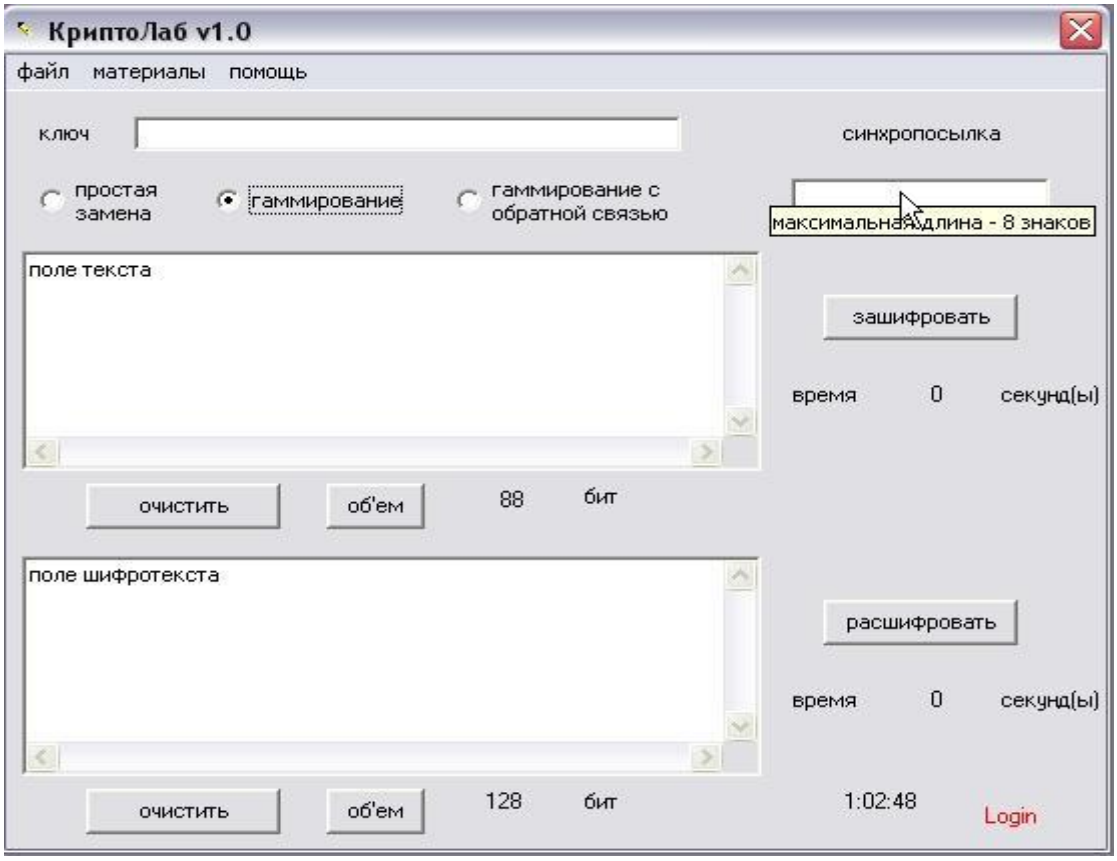


Рис. 3.9. Лабораторный комплекс

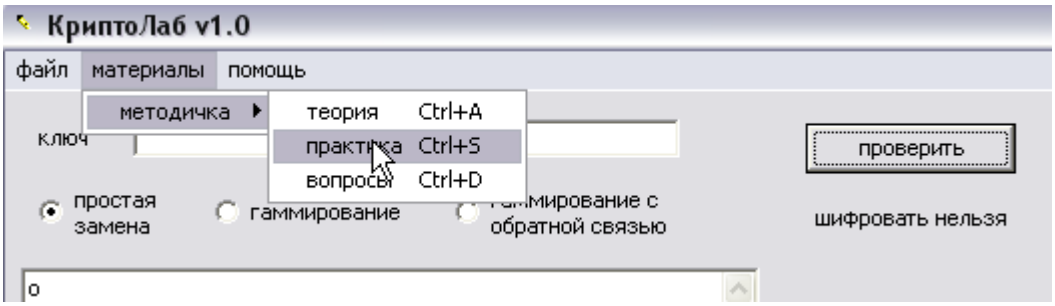


Рис. 3.10. Предоставленные материалы

Порядок выполнения работы

1. Ознакомится с теорией по стандарту ГОСТ 28147-89 (режимы шифрования, строение основного шага криптопреобразования, особенности использования таблицы замен).

2. Взять у преподавателя задание, состоящие из:

уникального ключа для шифрования/дешифрования текстовых сообщений, синхропосылки

3. Исследование стандарта шифрования

3.1 выбрать режим шифрования «простая замена»

3.2 ввести значение ключа

3.3 ввести тестовый текст (взять у преподавателя или пройти **материалы –> методичка –> практика –> тексты –> простая замена**

3.4 проверить кратность 8 байтовых блоков нажатием кнопки «проверить»

3.5 зашифровать текст нажатием соответствующей кнопки

3.6 очистить текстовое поле

3.7 расшифровать текст

3.8 проверить наличие ошибок

3.9 зафиксировать об’емы текстов и время за/расшифрования в таблицу

3.10 Проделать пункты 3.3-3.9 для оставшихся текстов

3.11 выбрать режим шифрования «гаммирование»

3.12 ввести тестовый текст (взять у преподавателя или пройти **материалы –> методичка –> практика –> тексты –> гаммирования**

3.13 ввести значения ключа и синхропосылки

3.14 зашифровать текст нажатием соответствующей кнопки

3.15 очистить текстовое поле

3.16 расшифровать текст

3.17 проверить наличие ошибок

3.18 зафиксировать об’емы текстов и время за/расшифрования в таблицу

3.19 Проделать пункты 3.14-3.18 для оставшихся текстов

3.20 проделать пункты 3.11-3.19 для режима «гаммирование с обратной связью»

3.21 Построить графики зависимости размера файла от времени его шифрования /расшифрования для всех режимов.

3.22 проверить изменение шифротекста при изменении ключа и синхропосылки

3.23 посмотреть как изменится текст после расшифрования если в шифротекст внести изменения (для всех режимов шифрования)

Таблица 2.3. Результаты измерения

Размер файла				
Время зашифрования				
Время расшифрования				

Содержание отчета

1 Цель работы.

2 Описание действий по каждому пункту.

3 Результаты проделанной работы (таблицы, графики, скриншоты).

4 Выводы.

Контрольные вопросы

1. Что такое симметричное шифрование, блочные коды?
2. Что представляет собой стандарт ГОСТ 28147-89 (длина ключа, размер входного блока, ключа, таблица замен)?
3. Что собой представляет архитектура данного стандарта?
4. Как устроен основной шага криптопреобразования?
5. Сколько циклов шифрования предусмотрено стандартом?
6. Что быстрее шифрование или расшифрование? Почему?
7. Какие режимы шифрования применяются в стандарте ГОСТ 28147-89?
8. Какие режимы быстрее при расшифровании? Почему?
9. Какие режимы лучше восстанавливают зашифрованную информацию при ошибке в одном символе? Почему?
10. С какой целью используется синхропосылка или вектор инициализации?

Лабораторная работа 4. Исследование Международного алгоритма симметричного шифрования AES

Интерфейс учебно-программного комплекса

Учебно-программный комплекс (в дальнейшем просто комплекс) был разработан на языке высокого уровня Visual Basic.

Главное окно

На рисунке 2.51 приведен интерфейс главного окна комплекса.

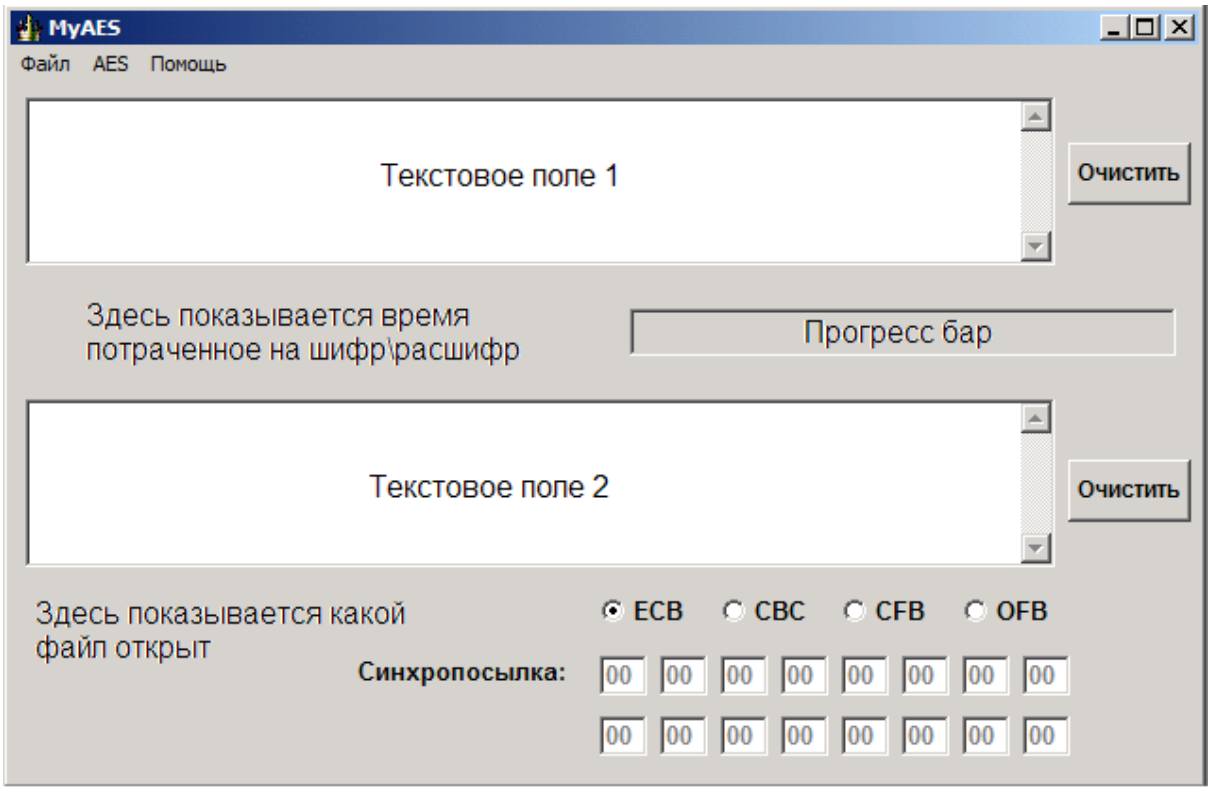


Рис. 4.1. Интерфейс главного окна комплекса

Главное окно состоит из:

1. Текстовое поле 1 – сюда вводится открытый текст, также если открывается файл с расширением txt, то его содержимое помещается сюда же.

2. Текстовое поле 2 – сюда помещается зашифрованный текст, также если открывается файл с расширением aes, то его содержимое помещается сюда же.

3. Прогресс бар – наглядно показывает время требуемое для шифрования \ расшифрования.

4. Кнопки “Очистить” – удаляют весь текст из текстовых полей.

5. Переключатели ECB, CBC, CFB, OFB – показывают, какой режим блочного шифрования используется.

6. Синхропосылка – задает синхропосылку для CBC, CFB и OFB.

В главном окне и происходит шифрование и расшифрование. Также в нем можно выбрать режимы симметричного шифрования (ECB, CBC, CFB, OFB). Про них сказано ниже.

Пункт меню “Файл”

На рисунке 2.12 приведен пункт меню “Файл”:

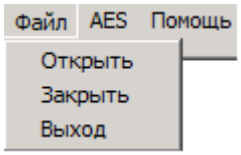


Рис. 4.2. Пункт меню “Файл”

Пункт меню “Файл” содержит следующие опции:

1. Открыть – открывает файл (*.txt, *.aes), но не больше 32kB для текстовых файлов и 64кВ для файлов типа aes.

2. Закрыть – закрывает выбранный файл.

3. Выход – завершает работу комплекса.

Пункт меню “AES”

На рисунке 2.13 приведен пункт меню “AES”:

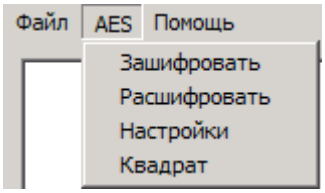


Рис. 4.3. Пункт меню “AES”

Пункт меню “AES” содержит следующие опции:

1. Зашифровать – шифрует текст, расположенный в текстовом поле 1, и помещает его в текстовое поле 2.

2. Расшифровать - расшифровывает текст, расположенный в текстовом поле 2, и помещает его в текстовое поле 1.

3. Настройки – позволяет выбрать свой ключ шифрования, а также включить \ выключить режим логирования. (рисунок 4.4)

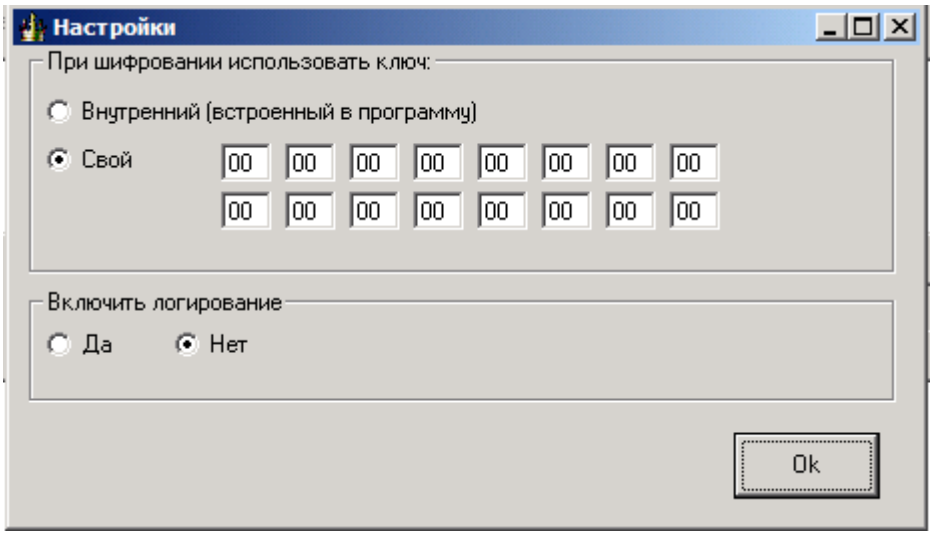


Рис. 4.4. Окно “Настройки”

В этом окне вы можете ввести свой ключ шифрования (например из задания по лабораторной работе) или же использовать внутренний. При использовании внутреннего ключа шифрования вы можете шифровать свои сообщения и отправлять их кому-либо. Причем получателю не обязательно знать ключ, главное чтобы у него была данная программа.

Здесь же вы можете включить логирование. При включенном логировании в папке программы создается Encrypt.txt (Decrypt.txt), в зависимости от того, что вы делаете шифруете или расшифровываете. В этом файле содержатся все преобразования, происходящие с открытым текстом. Пример такого файла приведен на рисунке 4.5:

```
Зашифрование
Входной блок CFF0EEE2E5F0EAE001010101010101010
Раундовый ключ: 00000000000000000000000000000000
Состояние после сложения с ключом: CFF0EEE2E5F0EAE001010101010101010
Раунд №1
Состояние после SubBytes: 8A8C2898D98C87E17C7C7C7C7C7C7C7C
Состояние после ShiftRows: 8A8C7C7CD97C7C987C7C28E17C8C877C
Состояние после MixColumns: 80717A8DC93DEE5BB51D68098C916177
Раундовый ключ: 626363636263636362636363626363636
Состояние после сложения с ключом: E21219EEAB5E8D38D77E0B6AEEF20214
Раунд №2
Состояние после SubBytes: 98C9D42862585D070EF32B02288977FA
Состояние после ShiftRows: 98582BFA62F377280E89D40728C95D02
Состояние после MixColumns: 12AF832F952E07724F673D414F445DE8
Раундовый ключ: 9B9898C9F9FBFBAA9B9898C9F9FBFBAA
Состояние после сложения с ключом: 89371BE66CD5FCD8D4FFA588B6BFA642
Раунд №3
Состояние после SubBytes: A79AAF8E5003B06148160 6C44E08242C
Состояние после ShiftRows: A703062C5016248E4808AF614E9AB0C4
Состояние после MixColumns: 7A87DCAF309E87C546D3A6BD5D6EF86B
Раундовый ключ: 909734506 96CCFFAF2F457330B0FAC99
Состояние после сложения с ключом: EA10E8FF59F2483FB427F18E566154F2
```

Рис. 4.5. Зашифрование слова “Проверка”

Логирование работает только в режиме ECB, и автоматически отключается при открытии файла или использовании другого режима.

4. Квадрат – простенькая реализация единственной атаки на AES, под названием квадрат. Реализована для сокращенной версии AES, для 4 раундов. В стандарте же их 10.

Окно Атака «Квадрат»

Смысл атаки “Квадрат” описан выше.

В этом окне (рисунок 2.16) можно просмотреть ключи для любого раунда. При открытии этого окна в поля “Реальный ключ” загружается ключ, заданный в настройках. При нажатии кнопок “Up” или ”Down” будет меняться номер раунда, а также поля “Реальный ключ”. Они будут равны тому раундовому ключу, который указан в строке “Номер раунда”. Нас будет интересовать ключ 4 раунда поэтому, лучше поставить значение поля “Номер раунда” в 4.

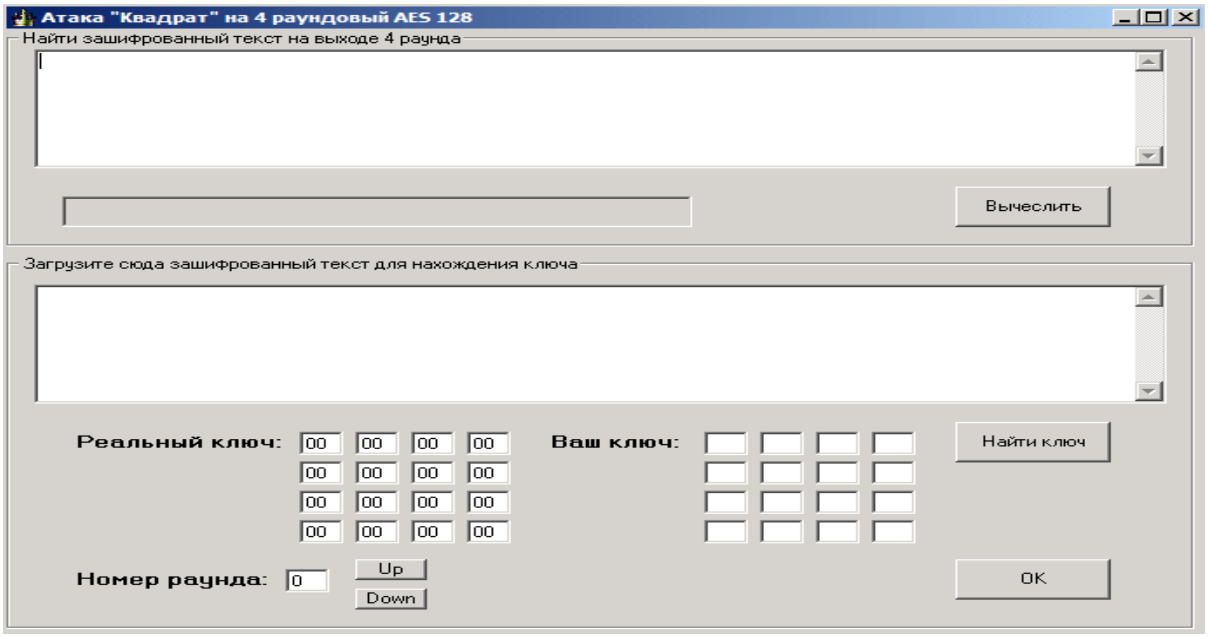


Рис. 4.6. Окно “Квадрат”

При нажатии кнопки “Вычислить” в текстовом поле 1 появляется зашифрованный L набор после 4 раунда. Прогресс бар наглядно показывает время требуемое для его вычисления. Пример L набора, зашифрованного ключом “00”{32}, показан на рисунке 4.7:

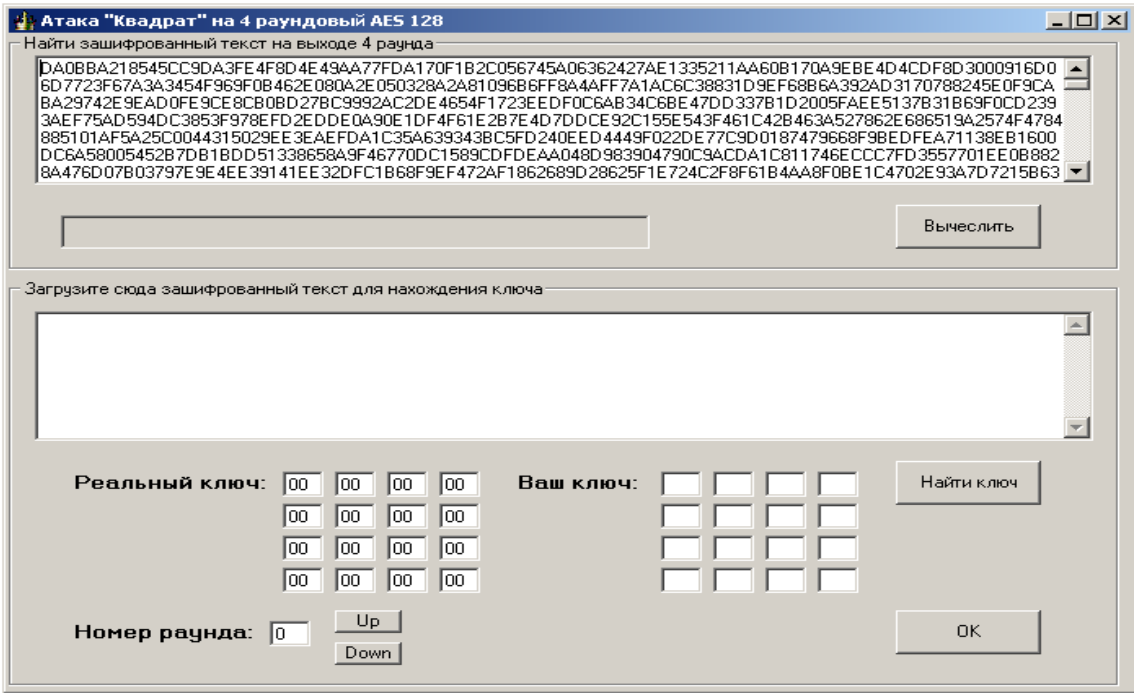


Рис. 4.7. Пример L набора, зашифрованного ключом “00”{32}

Этот текст нужно скопировать и вставить в текстовое поле 2.

При нажатии на кнопку “Найти ключ” начнется вычисление ключа 4 раунда на основе текста в текстовом поле 2. Прогресс бар наглядно показывает время, требуемое для его вычисления.

После вычисления ключа он появится в окошках “Ваш ключ”. Можно сравнить окна “Ваш ключ” и “Реальный ключ” и увидеть, что они практически совпадают (рисунок 4.8).

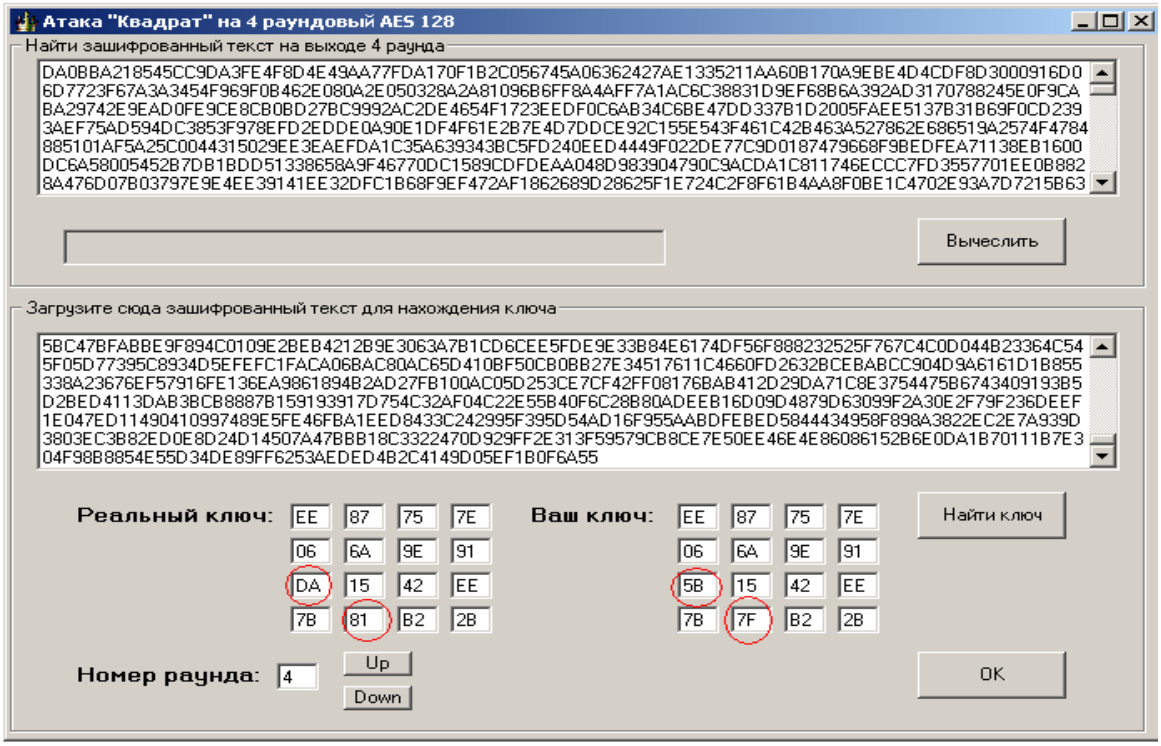


Рис. 4.8. Пример вычисления раундового ключа

Режимы ECB, CBC, CFB, OFB

Для различных ситуаций, встречающихся на практике, разработано значительное количество режимов шифрования.

Режим ECB (Electronic Code Book – режим электронной кодовой книги)

В режиме ECB каждый блок открытого текста заменяется блоком шифротекста. А так как один и тот же блок открытого текста заменяется одним и тем же блоком шифротекста, теоретически возможно создать кодовую книгу блоков открытого текста и соответствующих шифротекстов. Но если размер блока составляет n бит, кодовая книга будет состоять из 2n записей.

Режим ECB - простейший режим шифрования. Все блоки открытого текста шифруются независимо друг от друга. Это важно для шифрованных файлов с произвольным доступом, например, файлов баз данных. Если база данных зашифрована в режиме ECB, любая запись может быть добавлена, удалена, зашифрована или расшифрована независимо от любой другой записи (при условии, что каждая запись состоит из целого числа блоков шифрования). Кроме того, обработка может быть параллельной: если используются несколько шифровальных процессоров, они могут шифровать или расшифровывать различные блоки независимо друг от друга. Режим ECB показан на рисунке 2.19:

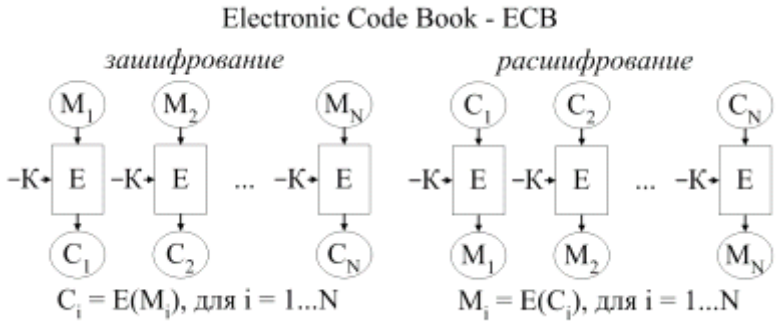


Рис. 4.9. Режим ECB

К недостаткам режима ECB можно отнести то обстоятельство, что если у криптоаналитика есть открытый текст и шифротекст нескольких сообщений, он может, не зная ключа, начать составлять шифровальную книгу. В большинстве реальных ситуаций фрагменты сообщений имеют тенденцию повторяться. В различных сообщениях могут быть одинаковые битовые последовательности, В сообщениях, которые, подобно электронной почте, создаются компьютером, могут быть периодически повторяющиеся структуры. Сообщения могут быть высоко избыточными или содержать длинные строки нулей или пробелов.

К достоинствам режима ECB можно отнести возможность шифрования нескольких сообщений одним ключом без снижения надежности. По существу, каждый, блок можно

рассматривать как отдельное сообщение, шифрованное тем же самым ключом. При расшифровании ошибки в символах шифротекста ведут к некорректному расшифрованию соответствующего блока открытого текста, однако не затрагивают остальной открытый текст. Но если бит шифротекста случайно потерян или добавлен, весь последующий шифротекст будет дешифрован некорректно, если только для выравнивания границ блоков не используется какое-нибудь выравнивание по границам блока. Большинство сообщений не делятся точно на n - битовые блоки шифрования – в конце обычно оказывается укороченный блок. Однако режим ECB требует использовать строго n - битовые блоки. Для решения этой проблемы используют дополнение (padding), Чтобы создать полный блок, последний блок дополняют некоторым стандартным шаблоном - нулями, единицами, чередующимися нулями и единицами.

Режим CBC (Ciphertext Block Chaining – режим сцепления блоков шифротекста)

Сцепление добавляет в блочный шифр механизм обратной связи: результаты шифрования предыдущих блоков влияют на шифрование текущего блока, т.е. каждый блок используется для модифицирования шифрования следующего блока. Каждый блок шифротекста зависит не только от шифруемого блока открытого текста, но и от всех предыдущих блоков открытого текста.

В режиме сцепления блоков шифротекста перед шифрованием над открытым текстом и предыдущим блоком шифротекста выполняется операция XOR. Процесс шифрования в режиме CBC. Когда блок открытого текста зашифрован, полученный шифротекст сохраняется в регистре обратной связи. Следующий блок открытого текста перед шифрованием подвергается операции XOR с содержимым регистра обратной связи. Результат операции XOR используется как входные данные для следующего этапа процедуры шифрования. Полученный шифротекст снова сохраняется в регистре обратной связи, чтобы подвергнуться операции XOR вместе со следующим блоком открытого текста, и так до конца сообщения. Шифрование каждого блока зависит от всех предыдущих блоков.

Расшифрование выполняется в обратном порядке. Блок шифротекста расшифровывается обычным путем, но сохраняется в регистре обратной связи. Затем следующий блок расшифровывается и подвергается операции XOR с содержимым регистра обратной связи. Теперь следующий блок шифротекста сохраняется в регистре обратной связи и т.д. до конца сообщения.

На рисунке 4.10 показан режим CBC (IV – синхропосылка):

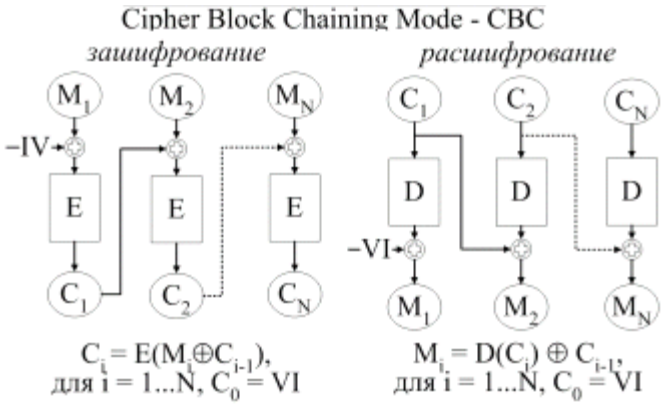


Рис. 4.40. Режим CBC

При шифровании в режиме CBC одинаковые блоки открытого текста превращаются в различающиеся друг от друга блоки шифротекста только в том случае, если различались какие-то предшествующие блоки открытого текста. Однако при шифровании двух 48 идентичных сообщений создается один и тот же шифротекст. Хуже того, два одинаково начинающихся сообщения будут шифроваться одинаково вплоть до первого различия.

Чтобы избежать этого, можно зашифровать в первом блоке какие-то произвольные данные. Этот блок случайных данных называют вектором инициализации (ВИ) (Initialization Vector, IV, русский термин - синхропосылка), инициализирующей переменной или начальным значением сцепления. Вектор ВИ не имеет какого-то смыслового значения, он используется только для того, чтобы сделать каждое сообщение уникальным. Когда получатель расшифровывает этот блок, он использует его только для заполнения регистра обратной связи. В качестве вектора ВИ удобно использовать метку времени, либо какие-то случайные биты.

Если используется вектор инициализации, сообщения с идентичным открытым текстом после шифрования превращаются в сообщения с разными шифротекстами. Следовательно, злоумышленник не может попытаться повторить блок, и создание шифровальной книги затруднится. Хотя для каждого сообщения, шифруемого одним и тем же ключом, рекомендуется выбирать уникальный вектор ВИ, это требование необязательное. Вектор ВИ не обязательно хранить в секрете, его можно передавать открыто - вместе с шифротекстом.

Режим CFB (Ciphertext Feedback – обратная связь по шифротексту)

В режиме CBC начать шифрование до поступления полного блока данных невозможно. Для некоторых сетевых приложений это создает проблемы. Например, в защищенном сетевом окружении терминал должен иметь возможность передавать хосту каждый символ сразу после ввода. Если же данные нужно обрабатывать блоками в несколько байт, режим CBC просто не работает.

На рисунке 4.11 показан режим CFB (IV – синхропосылка):

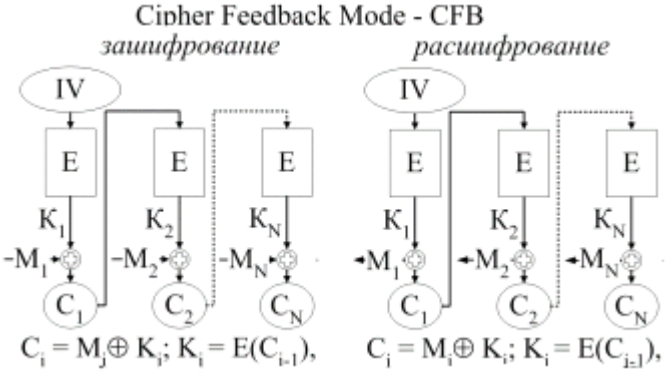


Рис. 4.11. Режим CFB2

Как и в режиме CBC, первоначально очередь заполнена вектором инициализации ВИ. Очередь шифруется, затем выполняется операция XOR над *n* старшими (крайними левыми) битами результата и первым *n* -битовым символом открытого текста. В результате появляется первый *n* -битовый символ шифротекста. Теперь этот символ можно передать. Кроме того, полученные *n* битов попадают в очередь на место *n* младших битов, а все остальные биты сдвигаются на *n* позиций влево. Предыдущие *n* старших битов отбрасываются. Затем точно также шифруется следующие *n* битов открытого текста. Расшифрование выполняется в обратном порядке. Обе стороны - шифрующая и расшифровывающая - использует блочный алгоритм в режиме шифрования. Как и режим CBC, режим CFB сцепляет символы открытого текста с тем, чтобы шифротекст зависел от всего предыдущего открытого текста.

Для инициализации процесса шифрования в режиме CFB в качестве входного блока алгоритма можно использовать вектор инициализации ВИ. Как и в режиме CBC, хранить в тайне вектор ВИ не нужно. Однако вектор ВИ должен быть уникальным. (В отличие от режима CBC, где уникальность вектора ВИ необязательна, хотя и желательна). Если вектор ВИ в режиме CFB не уникален, криптоаналитик может восстановить соответствующий открытый текст. Вектор инициализации должен меняться в каждом сообщении. Например, вектором ВИ может служить порядковый номер, возрастающий в каждом новом сообщении и не повторяющийся все время жизни ключа. Если данные шифруются с целью последующего хранения, вектор ВИ может быть функцией индекса, используемого для поиска данных.

Режим OFB (Output Feedback – режим обратной связи по выходу)

Режим OFB представляет собой метод использования блочного шифра в качестве синхронного потокового шифра. Этот режим подобен режиму CFB.

Блочный алгоритм работает в режиме шифрования как на шифрующей, так и на расшифровывающей сторонах. Такую обратную связь иногда называют внутренней, поскольку механизм обратной связи не зависит ни от потока открытого текста, ни от потока шифротекста.

К достоинствам режима OFB относится то, что большую часть работы можно выполнить оффлайново, даже когда открытого текста сообщения еще вовсе не существует. Когда, наконец, сообщение поступает, для создания шифротекста над сообщением и выходом алгоритма необходимо выполнить операцию XOR.

В сдвиговый регистр OFB сначала надо загрузить вектор ВИ. Вектор должен быть уникальным, но сохранять его в тайне не обязательно.

На рисунке 2.22 показан режим OFB (IV – синхропосылка):

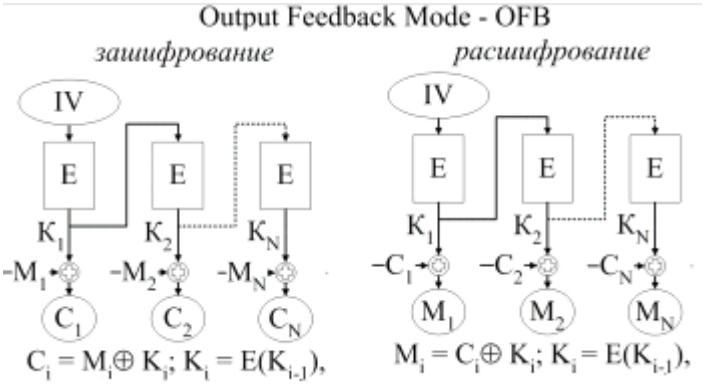


Рис. 4.12. Режим OFB

Анализ режима OFB показывает, что OFB целесообразно использовать только если разрядность обратной связи совпадает с размером блока.

В режиме OFB над гаммой и текстом выполняется операция XOR. Эта гамма, в конце концов, повторяется. Существенно, чтобы она не повторялась для одного и того же ключа, в противном случае секретность не обеспечивается ничем.

Порядок выполнения работы

Ознакомится с теорией по стандарту AES (из чего состоит раунд шифрования, как работают раундовые преобразования, с необходимым математическим аппаратом).

Взять у преподавателя задание, состоящие из:

- уникального ключа для шифрования/дешифрования текстовых сообщений
- набора файлов разного размера для исследования стандарта
- файла для исследования режимов шифрования
- вектора инициализации
- файла для проведения атаки «Квадрат»

Исследование стандарт шифрования

1. Открыть окно «Настройки» (AES>Настройки>), установить переключатель в положение «Свой ключ» и ввести посимвольное значение ключа (32 символа, 16 ячеек). Нажать «ОК».

2. В главном окне поставить переключатель в положение ECB.

3. Открыть файл из папки программы (Файл >Открыть)

4. Зашифровать и расшифровать его. (AES > Зашифровать (Расшифровать)).

5. Проделать пункты 3.2-3.4 для оставшихся трех файлов.

6. По полученным результатам заполнить таблицу

Размер файла				
Время зашифрования				
Время расшифрования				

7. Построить графики зависимости шифрования \ расшифрования от времени.

8. По полученным данным оценить примерное время взлома стандарта AES с помощью всего перебора ключей. Построить график.

\Пример:

На расшифрование 5кВ (5120 байт) текста на Athlon 2600+ тратится 9с. Поэтому один блок (16 байт) программа расшифровывает за 9/(5120/16) = (9*16)/5120 = 28 мс. Таким образом, на перебор 1 ключа, а это 128 бит, потребуется 28 мс.

За 1 секунду программа способна перебрать 4571 бит или примерно 35 вариантов ключей.

За год непрерывной работы программа переберет (35*60*60*24*365) = 1 103 760 000 ключей.

Если поделить 2^{128} на 1 103 760 000 , то получим $3.08*10^{29}$ лет непрерывной работы программы.

Исследование быстродействия различных режимов шифрования

1. Поставить переключатель в режим CBC.

2. Открыть файл из папки программы (Файл >Открыть)

3. Зашифровать и расшифровать его (AES > Зашифровать (Расшифровать))

4. Повторить пункт 4.2-4.3 для оставшихся трех файлов (10, 15, 30 кБ)

5. Поставить переключатель в режим CFB.

6. Повторить пункты 4.2 – 4.3.

7. Поставить переключатель в режим OFB.

8. Повторить пункты 4.2 – 4.3

9. По полученным результатам заполнить таблицу:

Размер файла				
ECB время заш/расш				
CBC время заш/расш				
CFB время заш/расш				
OFB время заш/расш				

10. Построить на одном графике зависимость шифрования \ расшифрования от времени.

11. Для каждого режима загрузить еще раз файл (любой). Зашифровать его. В текстовом поле 2 изменить какой-нибудь символ. Нажать AES > Расшифровать.

Исследование свойств различных режимов шифрования

Загрузить файл «Свойства режимов».

Поочередно зашифровать текстовое сообщение во всех четырех режимах.

Сравнить полученные шифротексты и сделать выводы.

Исследование пораундовой работы алгоритма

1 Открыть любой из предоставленных для работы файлов

2. Включить функцию логирования (AES>Настройки>Включить логирование>Да).

3. В главном окне поставить переключатель в положение ECB.

4. Зашифровать и расшифровать.

5. Открыть с помощью текстового редактора файлы Encrypt.txt и Decrypt.txt из папки программы

6. Проанализировать полученный результат.

Знакомство с атакой «Квадрат»

1. В главном окне в текстовое поле 1 загрузить файл «Атака Квадрат» (Файл>Открыть)

2. Открыть окно Атака «Квадрат» (AES > Квадрат)

3. С помощью переключателя “Номер раунда” установить 4 раундовый ключ.

4. Нажать кнопку “вычислить”.

5. Поместить текст из поля 1 в поле 2.

6. Нажать кнопку найти ключ.

7. Сравнить значения текстовых полей “Ваш ключ” и значения текстовых полей “Реальный ключ”.

8. Сделать выводы.

Содержание отчета

1. Цель работы.

2. Описание действий по каждому пункту.

3. Результаты проделанной работы (таблицы, графики, расчетная часть).

4. Выводы.

Контрольные вопросы

1. Что такое симметричное шифрование?

2. Что представляет собой стандарт AES (длина ключа, размер входного блока)?

3. Какой алгоритм выбран в качестве стандарта AES?

4. Что собой представляет архитиктура данного стандарта?

5. Из чего состоит один раунд?

6. Сколько раундов шифрования предусмотрено стандартом?

7. Что быстрее шифрование или разшифрование? Почему?

8. Какие режимы шифрования применяются в стандарте AES?

9. Какие режимы быстрее при разшифровании? Почему?

10. Какие режимы лучше восстанавливают зашифрованную информацию при ошибке в одном символе? Почему?

11. С какой целью используется синхропосылка или вектор инициализации?

12. Что представляет собой атака Квадрат? Какие ее особенности?

1. Ознакомится с теорией по стандарту AES (из чего состоит раунд шифрования, как работают раундовые преобразования, с необходимым математическим аппаратом).

2. Взять у преподавателя задание, состоящие из:

- уникального ключа для шифрования/дешифрования текстовых сообщений
- набора файлов разного размера для исследования стандарта
- файла для исследования режимов шифрования
- вектора инициализации
- файла для проведения атаки «Квадрат»

3. Исследование стандарт шифрования

3.1. Открыть окно «Настройки» (AES>Настройки>), установить переключатель в положение «Свой ключ» и ввести посимвольное значение ключа (32 символа, 16 ячеек). Нажать «ОК».

3.2. В главном окне поставить переключатель в положение ECB.

3.3. Открыть файл из папки программы (Файл >Открыть)

3.4. Зашифровать и расшифровать его. (AES > Зашифровать (Расшифровать)).

3.5. Проделать пункты 3.2-3.4 для оставшихся трех файлов.

3.5. По полученным результатам заполнить таблицу

Размер файла				
Время зашифрования				
Время расшифрования				

3.6. Построить графики зависимости шифрования \ расшифрования от времени.

3.7. По полученным данным оценить примерное время взлома стандарта AES с помощью всего перебора ключей. Построить график.

Пример:

На расшифрование 5кВ (5120 байт) текста на Athlon 2600+ тратится 9с. Поэтому один блок (16 байт) программа расшифровывает за 9/(5120/16) = (9*16)/5120 = 28 мс. Таким образом, на перебор 1 ключа, а это 128 бит, потребуется 28 мс.

За 1 секунду программа способна перебрать 4571 бит или примерно 35 вариантов ключей.

За год непрерывной работы программа переберет (35*60*60*24*365) = 1 103 760 000 ключей.

Если поделить 2^{128} на 1 103 760 000 , то получим $3.08*10^{29}$ лет непрерывной работы программы.

3.8. Сделать выводы

Исследование быстродействия различных режимов шифрования

1. Поставить переключатель в режим CBC.

2. Открыть файл из папки программы (Файл >Открыть)

3. Зашифровать и расшифровать его (AES > Зашифровать (Расшифровать))

4. Повторить пункт 4.2-4.3 для оставшихся трех файлов (10, 15, 30 кБ)

5. Поставить переключатель в режим CFB.

6. Повторить пункты 4.2 – 4.3.

7. Поставить переключатель в режим OFB.

8. Повторить пункты 4.2 – 4.3

9. По полученным результатам заполнить таблицу:

Размер файла				
ECB время заш/расш				
CBC время заш/расш				
CFB время заш/расш				
OFB время заш/расш				

10. Построить на одном графике зависимость шифрования \ расшифрования от времени.

11. Для каждого режима загрузить еще раз файл (любой). Зашифровать его. В текстовом поле 2 изменить какой-нибудь символ. Нажать AES > Расшифровать.

12. Сделать выводы.

Исследование свойств различных режимов шифрования

Загрузить файл «Свойства режимов».

Поочередно зашифровать текстовое сообщение во всех четырех режимах.

Сравнить полученные шифротексты и сделать выводы.

Исследование пораундовой работы алгоритма

1 Открыть любой из предоставленных для работы файлов

2. Включить функцию логирования (AES>Настройки>Включить логирование>Да).

3. В главном окне поставить переключатель в положение ECB.

4. Зашифровать и расшифровать.

5. Открыть с помощью текстового редактора файлы Encrypt.txt и Decrypt.txt из папки программы

6. Проанализировать полученный результат.

Знакомство с атакой «Квадрат»

1. В главном окне в текстовое поле 1 загрузить файл «Атака Квадрат» (Файл>Открыть)

2. Открыть окно Атака «Квадрат» (AES > Квадрат)

3. С помощью переключателя “Номер раунда” установить 4 раундовый ключ.

4. Нажать кнопку “вычислить”.

5. Поместить текст из поля 1 в поле 2.

6. Нажать кнопку найти ключ.

7. Сравнить значения текстовых полей “Ваш ключ” и значения текстовых полей “Реальный ключ”.

8. Сделать выводы.

Содержание отчета

1. Цель работы.

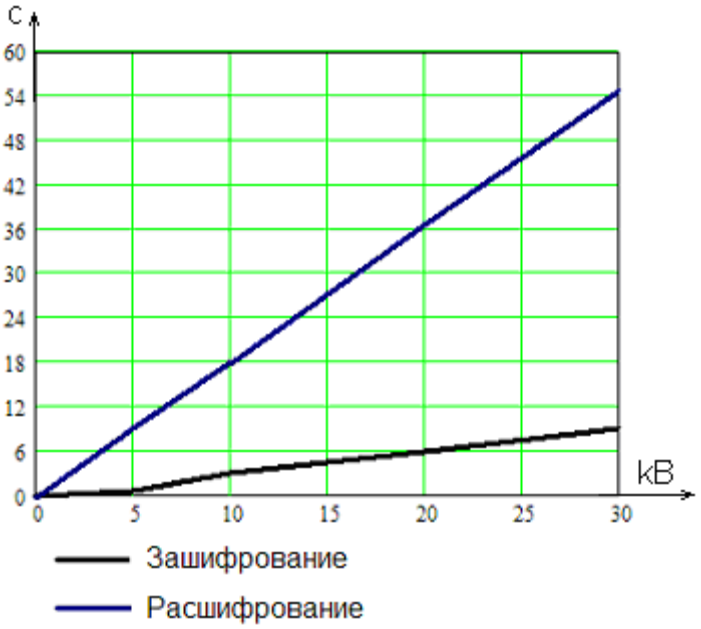
2. Описание действий по каждому пункту.

3. Результаты проделанной работы (таблицы, графики, расчетная часть).

4. Выводы.

Исследование стандарт шифрования

Размер файла	5kB	10kB	20kB	30kB
Время зашифрования, с	1	3	6	9
Время расшифрования, с	9	18	37	55



Зависимости размера файла от времени его шифрования \ расшифрования при проведении эксперимента на Athlon 2600+

Исследование быстродействия различных режимов шифрования

Размер файла	5kB	10kB	20kB	30kB
ECB время зашифрования	1	3	6	9
ECB время расшифрования	9	18	37	55
CBC время зашифрования	1	3	6	9
CBC время расшифрования	8	17	37	56
CFB время зашифрования	1	3	5	9
CFB время расшифрования	2	3	6	10
OFB время зашифрования	1	3	6	9
OFB время расшифрования	2	3	7	9

Зашифруем разными режимами слово пример, затем в шифротексте изменим 1 символ, получим:

	Исходный текст	Исходный шифротекст	Измененный шифротекст	Результат
ECB	пример	63891BCDBE14D6E7 B827C0AF68EF5419	63891BCDBE14D6E7 B827C0AF68EF5410	ЛМЎ!ня«m;5J9w
CBC	пример	0000000000000000 0000000000000638 91BCDBE14D6E7B82 7C0AF68EF5419	1000000000000000 0000000000000638 91BCDBE14D6E7B82 7C0AF68EF5419	яример
CFB	пример	0000000000000000 0000000000000891 9A3380A7A2D3A894D FB58CB352A2F	0000000000000000 0000000000000689 19A3380A7A2D3A894D FB58CB352A2F	ЗЮSў”[ж<Яс-(s
OFB	пример	0000000000000000 0000000000000891 9A3380A7A2D3A894D FB58CB352A2F	0000000000001000 0000000000000891 9A3380A7A2D3A894D FB58CB352A2F	µZi'aIj·jOÕC&Ú

Исследование свойств различных режимов шифрования

Файл «Свойства режимов» содержит несколько раз повторяющееся слово. Например, *задание задание задание задание задание задание задание задание*.

Каждый режим при этом покажет свои особенности:

1. ECB

задание задание задание задание задание задание задание задание

D3D5451C8296E7E653A82652AED4E56A

D3D5451C8296E7E653A82652AED4E56A

D3D5451C8296E7E653A82652AED4E56A

A8F11E11A02FF140F0936291695512CA

2. CBC

00000000000000000000000000000000

D3D5451C8296E7E653A82652AED4E56A

F84E11957A1A02F67B97ED2A9DE04DD2

00D20F40BEE5D0C7EEF499531BEF8B8C

9BB124631613C483AE57E40BCD63C2AC

3. CFB

00000000000000000000000000000000

8109AF340262C91B6FAC1EB927DCCE0E

6D376DFF127D70059697E293142F5DA6

DBC4EAC4CA79A6139E944EB9B374DDAD

4128D7EF70BE71BB9F57B9D64DF4EC68

4. OFB

00000000000000000000000000000000

8109AF340262C91B6FAC1EB927DCCE0E

107559AABF0A7BF7F433F71ACD01689C

46EC128DE2351660B490505F601D5A93

A02767750D304B01733AEE7046216D8B

В данном примере синхропосыла равна 00000000000000000000000000000000

Исследование пораундовой работы алгоритма

Используем слово «пример»

<u>Зашифрование:</u>

Зашифрование

Входной блок EFF0E8ECE5F001010101010101010101

Раундовый ключ: 00000000000000000000000000000000

Состояние после сложения с ключом: EFF0E8ECE5F001010101010101010101

Раунд №1

Состояние после SubBytes: DF8C9BCED98C7C7C7C7C7C7C7C7C7C7C

Состояние после ShiftRows: DF8C7C7CD97C7CCE7C7C9B7C7C8C7C7C

Состояние после MixColumns: 2A242F729F6B14F79B4EA99B77878C8C

Раундовый ключ: 62636363626363636263636362636363

Состояние после сложения с ключом: 48474C11FD087794F92DCAF815E4EFEF

Раунд №2

Состояние после SubBytes: 52A029825430F52299D874415969DFDF

Состояние после ShiftRows: 523074DF54D8DF829969292259A0F541

Состояние после MixColumns: 5F71F0178607B4E49912C4B4FD47CB3C

Раундовый ключ: 9B9898C9F9FBFBAA9B9898C9F9FBFBAA

Состояние после сложения с ключом: C4E968DE7FFC4F4E028A5C7D04BC3096

Раунд №3

Состояние после SubBytes: 1C1E451DD2B0842F777E4AFFF2650490

Состояние после ShiftRows: 1CB04A90D27E041D7765452FF21E84FF

Состояние после MixColumns: 292993E5243F832D2B5DE9E7A6A6E572

Раундовый ключ: 90973450696CCFFAF2F457330B0FAC99

Состояние после сложения с ключом: B9BEA7B54D534CD7D9A9BED4ADA949EB

Раунд №4

Состояние после SubBytes: 56AE5CD5E3ED290E35D3AE4895D33BE9

Состояние после ShiftRows: 56EDAEE9E3D33BD535D35C0E95AE2948

Состояние после MixColumns: C797DC705DC6226756624CCCB9E1B1B3

Раундовый ключ: EE06DA7B876A1581759E42B27E91EE2B

Состояние после сложения с ключом: 2991060BDAAC37E623FC0E7EC7705F98

Раунд №5

Состояние после SubBytes: A5816F2B57919A8E26B0ABF3C651CF46

Состояние после ShiftRows: A591AB4657B0CF2B26516F8EC6819AF3

Состояние после MixColumns: 143CB342814D1FD05EBB2053669966B7

Раундовый ключ: 7F2E2B88F8443E098DDA7CBBF34B9290

Состояние после сложения с ключом: 6B1298CA790921D9D3615CE895D2F427

Раунд №6

Состояние после SubBytes: 7FC94674B601FD3566EF4A9B2AB5BFCC

Состояние после ShiftRows: 7F014ACCB6EFBF7466B546352AC9FD9B

Состояние после MixColumns: 7B6FA54996DDA0797BE800337224B467

Раундовый ключ: EC614B851425758C99FF09376AB49BA7

Состояние после сложения с ключом: 970EEECC82F8D5F5E217090418902FC0

Раунд №7

Состояние после SubBytes: 88AB284B134103E698F001F2AD6015BA

Состояние после ShiftRows: 884101BA13F0154B986028E6ADAB03F2

Состояние после MixColumns: 73B31EAC739C144645C6992C56170DBB

Раундовый ключ: 217517873550620BACAF6B3CC61BF09B

Состояние после сложения с ключом: 52C6092B46CC764DE969F210900CFD20

Раунд №8

Состояние после SubBytes: 00B401F15A4B38E31EF989CA60FE54B7

Состояние после ShiftRows: 004B89B75AF954F11EFE01E360B438CA

Состояние после MixColumns: E3A180B701BE03BAC719DC00F591E1A3

Раундовый ключ: 0EF903333BA9613897060A04511DFA9F

Состояние после сложения с ключом: ED5883843A176282501FD604A48C1B3C

Раунд №9

Состояние после SubBytes: 556AEC5F80F0AA1353C0F6F24964AFEB

Состояние после ShiftRows: 55F0F6EB80C0AF5F5364EC13496AAAF2

Состояние после MixColumns: BC447434B0AEE44AF5A7C15B748A61E4

Раундовый ключ: B1D4D8E28A7DB9DA1D7BB3DE4C664941

Состояние после сложения с ключом: 0D90ACD63AD35D90E8DC728538EC28A5

Раунд №10

Состояние после SubBytes: D76091F680664C609B86409707CE3406

Состояние после ShiftRows: D7664006808634F69BCE916007604C97

Раундовый ключ: B4EF5BCB3E92E21123E951CF6F8F188E

Состояние после сложения с ключом: 63891BCDBE14D6E7B827C0AF68EF5419

Расшифрование

Разшифрование

Входной блок 63891BCDBE14D6E7B827C0AF68EF5419

Раундовый ключ: B4EF5BCB3E92E21123E951CF6F8F188E

Состояние после сложения с ключом: D7664006808634F69BCE916007604C97

Раунд №1

Состояние после invShiftRows: D76091F680664C609B86409707CE3406

Состояние после invSubBytes: 0D90ACD63AD35D90E8DC728538EC28A5

Раундовый ключ: B1D4D8E28A7DB9DA1D7BB3DE4C664941

Состояние после сложения с ключом: BC447434B0AEE44AF5A7C15B748A61E4

Состояние после invMixColumns: 55F0F6EB80C0AF5F5364EC13496AAAF2

Раунд №2

Состояние после invShiftRows: 556AEC5F80F0AA1353C0F6F24964AFEB

Состояние после invSubBytes: ED5883843A176282501FD604A48C1B3C

Раундовый ключ: 0EF903333BA9613897060A04511DFA9F

Состояние после сложения с ключом: E3A180B701BE03BAC719DC00F591E1A3

Состояние после invMixColumns: 004B89B75AF954F11EFE01E360B438CA

Раунд №3

Состояние после invShiftRows: 00B401F15A4B38E31EF989CA60FE54B7

Состояние после invSubBytes: 52C6092B46CC764DE969F210900CFD20

Раундовый ключ: 217517873550620BACAF6B3CC61BF09B

Состояние после сложения с ключом: 73B31EAC739C144645C6992C56170DBB

Состояние после invMixColumns: 884101BA13F0154B986028E6ADAB03F2

Раунд №4

Состояние после invShiftRows: 88AB284B134103E698F001F2AD6015BA

Состояние после invSubBytes: 970EEECC82F8D5F5E217090418902FC0

Раундовый ключ: EC614B851425758C99FF09376AB49BA7

Состояние после сложения с ключом: 7B6FA54996DDA0797BE800337224B467

Состояние после invMixColumns: 7F014ACCB6EFBF7466B546352AC9FD9B

Раунд №5

Состояние после invShiftRows: 7FC94674B601FD3566EF4A9B2AB5BFCC

Состояние после invSubBytes: 6B1298CA790921D9D3615CE895D2F427

Раундовый ключ: 7F2E2B88F8443E098DDA7CBBF34B9290

Состояние после сложения с ключом: 143CB342814D1FD05EBB2053669966B7

Состояние после invMixColumns: A591AB4657B0CF2B26516F8EC6819AF3

Раунд №6

Состояние после invShiftRows: A5816F2B57919A8E26B0ABF3C651CF46

Состояние после invSubBytes: 2991060BDAAC37E623FC0E7EC7705F98

Раундовый ключ: EE06DA7B876A1581759E42B27E91EE2B

Состояние после сложения с ключом: C797DC705DC6226756624CCCB9E1B1B3

Состояние после invMixColumns: 56EDAEE9E3D33BD535D35C0E95AE2948

Раунд №7

Состояние после invShiftRows: 56AE5CD5E3ED290E35D3AE4895D33BE9

Состояние после invSubBytes: B9BEA7B54D534CD7D9A9BED4ADA949EB

Раундовый ключ: 90973450696CCFFAF2F457330B0FAC99

Состояние после сложения с ключом: 292993E5243F832D2B5DE9E7A6A6E572

Состояние после invMixColumns: 1CB04A90D27E041D7765452FF21E84FF

Раунд №8

Состояние после invShiftRows: 1C1E451DD2B0842F777E4AFFF2650490

Состояние после invSubBytes: C4E968DE7FFC4F4E028A5C7D04BC3096

Раундовый ключ: 9B9898C9F9FBFBAA9B9898C9F9FBFBAA

Состояние после сложения с ключом: 5F71F0178607B4E49912C4B4FD47CB3C

Состояние после invMixColumns: 523074DF54D8DF829969292259A0F541

Раунд №9

Состояние после invShiftRows: 52A029825430F52299D874415969DFDF

Состояние после invSubBytes: 48474C11FD087794F92DCAF815E4EFEF

Раундовый ключ: 62636363626363636263636362636363

Состояние после сложения с ключом: 2A242F729F6B14F79B4EA99B77878C8C

Состояние после invMixColumns: DF8C7C7CD97C7CCE7C7C9B7C7C8C7C7C

Раунд №10

Состояние после invShiftRows: DF8C9BCED98C7C7C7C7C7C7C7C7C7C7C

Состояние после invSubBytes: EFF0E8ECE5F001010101010101010101

Раундовый ключ: 00000000000000000000000000000000

Состояние после сложения с ключом: EFF0E8ECE5F001010101010101010101

Лабораторная работа 5. Исследование алгоритма поточного шифрования RC4

Целью данной лабораторной работы является изучение алгоритма RC4, рассмотрение его слабых и сильных сторон, определение в каких продуктах и каким образом он используется, а так же знакомство с программной реализацией на языке C++, с использованием программного обеспечения Borland C++ Builder 6.

RC4 (англ. Rivest Cipher 4 или англ. Ron's Code, также известен как ARCFOUR или ARC4 (англ. Alleged RC4)) — это потоковый шифр, широко применяющийся в различных системах защиты информации в компьютерных сетях (например, в протоколах SSL и TLS, алгоритме безопасности беспроводных сетей WEP, для шифрования паролей).

Шифр разработан компанией RSA Security и для его использования требуется лицензия.

Алгоритм RC4 строится, как и любой потоковый шифр на основе параметризованного ключом генератора псевдослучайных битов с равномерным распределением. Длина ключа обычно составляет от 5 до 64 байт. Максимальная длина ключа 256 байт.

Основные преимущества шифра — высокая скорость работы и переменный размер ключа. RC4 довольно уязвим, если используются не случайные или связанные ключи, один ключевой поток используется дважды. Эти факторы, а также способ использования могут сделать криптосистему небезопасной (например, WEP).

Потоковый шифр RC4 был создан Роном Ривестом из RSA Security в 1987 году. Хотя официально сокращение обозначает Rivest Cipher 4, его часто считают сокращением от Ron's Code.

Шифр являлся коммерческой тайной, но в сентябре 1994 года его описание было анонимно отправлено в рассылку Cypherpunks. Вскоре описание RC4 было опубликовано в ньюс-группе sci.crypt. Именно оттуда исходный код попал на множество сайтов в сети Интернет. Опубликованный шифр давал те же шифротексты на выходе, какие давал подлинный RC4. По-видимому, данный текст был получен в результате анализа исполняемого кода. Опубликованный шифр совместим с имеющимися продуктами, использующими RC4, а некоторые участники телеконференции, имевшие, по их словам, доступ к исходному коду RC4, подтвердили идентичность алгоритмов при различиях в обозначениях и структуре программы.

Поскольку данный алгоритм известен, он более не является коммерческой тайной. Однако, название «RC4» является торговой маркой компании RSA. Поэтому иногда шифр

называют «ARCFOUR» или «ARC4» (имея ввиду Alleged RC4 — предполагаемый RC4, поскольку RSA официально не опубликовала алгоритм), чтобы избежать возможных претензий со стороны владельца торговой марки.

Шифр RC4 применяется в некоторых широко распространённых стандартах и протоколах шифрования таких, как WEP, WPA и TLS.

Главными факторами, способствовавшими широкому применению RC4, были простота его аппаратной и программной реализации, а также высокая скорость работы алгоритма в обоих случаях.

В США длина ключа для использования внутри страны рекомендуется равной 128 битов, но соглашение, заключённое между Software Publishers Association (SPA) и правительством США даёт RC4 специальный статус, который означает, что разрешено экспортировать шифры длиной ключа до 40 бит. 56-битные ключи разрешено использовать заграничным отделениям американских компаний.

Описание алгоритма

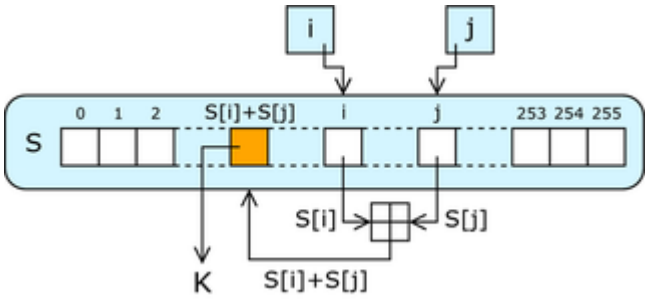


Рис. 5.1. Генератор ключевого потока RC4

Ядро алгоритма состоит из функции генерации ключевого потока. Эта функция генерирует последовательность битов (ki), которая затем объединяется с открытым текстом (mi) посредством суммирования по модулю два. Так получается шифрограмма (ci):

$$c_i = m_i \oplus k_i.$$

Расшифровка заключается в регенерации этого ключевого потока (ki) и сложении его и шифрограммы (ci) по модулю два. В силу свойств суммирования по модулю два на выходе мы получим исходный незашифрованный текст (mi):

$$m_i = c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i.$$

Другая главная часть алгоритма — функция инициализации, которая использует ключ переменной длины для создания начального состояния генератора ключевого потока.

RC4 — фактически класс алгоритмов, определяемых размером его блока. Этот параметр n является размером слова для алгоритма. Обычно, n = 8, но в целях анализа можно уменьшить его. Однако для повышения безопасности необходимо увеличить эту величину. Внутреннее состояние RC4 представляется в виде массива слов размером 2n и двух

счетчиков, каждый размером в одно слово. Массив известен как S-бокс, и далее будет обозначаться как S. Он всегда содержит перестановку 2n возможных значений слова. Два счетчика обозначены через i и j.

Алгоритм инициализации RC4 приведен ниже. Этот алгоритм также называется алгоритмом ключевого расписания (Key-Scheduling Algorithm or KSA). Этот алгоритм использует ключ, сохраненный в Key, и имеющий длину l байт. Инициализация начинается с заполнения массива S, далее этот массив перемешивается путем перестановок определяемых ключом. Так как только одно действие выполняется над S, то должно выполняться утверждение, что S всегда содержит все значения кодового слова.

Начальное заполнение массива:

for i = 0 to 2n − 1

S[i] = i

Скремблирование:

j = 0

for i = 0 to 2n − 1

j = (j + S[i] + Key[i mod l]) mod 2n

Перестановка (S[i], S[j])

Генератор ключевого потока RC4 переставляет значения, хранящиеся в S, и каждый раз выбирает различное значение из S в качестве результата. В одном цикле RC4 определяется одно n-битное слово K из ключевого потока, которое в последующем суммируется с исходным текстом для получения зашифрованного текста. Эта часть алгоритма называется генератором псевдослучайной последовательности (Pseudo-Random Generation Algorithm or PRGA).

Инициализация:

i = 0

j = 0

Цикл генерации:

i = (i + 1) mod 2n

j = (j + S[i]) mod 2n

Перестановка (S[i], S[j])

Результат: K = S[(S[i] + S[j]) mod 2n]

Безопасность

В отличие от современных шифров (таких, как в eSTREAM), RC4 не использует отдельной оказии (nonce) наряду с ключом. Это значит, что если один ключ должен использоваться в течение долгого времени для шифрования нескольких потоков, сама

криптосистема, использующая RC4, должна комбинировать оказию и долгосрочный ключ для получения потокового ключа для RC4. Один из возможных выходов — генерировать новый ключ для RC4 с помощью хэш-функции от долгосрочного ключа и оказии. Однако, многие приложения, использующие RC4, просто конкатенируют ключ и оказию. Из-за этого и слабого расписания ключей, используемого в RC4, приложение может стать уязвимым.

Здесь будут рассмотрены некоторые атаки на шифр и методы защиты от них.

Манипуляция битами

Шифр RC4 крайне уязвим к манипуляции битами, если он не реализован верным образом. И поэтому он был признан устаревшим многими софтверными компаниями, такими как Microsoft. Например, в .NET Framework от Microsoft отсутствует реализация RC4.

Исследования Руза и восстановление ключа из перестановки

В 1995 году Андрю Руз (Andrew Roos) экспериментально пронаблюдал, что первый байт ключевого потока коррелирован с первыми тремя байтами ключа, а первые несколько байт перестановки после алгоритма расписания ключей (KSA) коррелированны с некоторой линейной комбинацией байт ключа. Эти смещения не были доказаны до 2007 года, когда Пол, Рафи и Мэйтрэ доказали коррелированность ключа и ключевого потока. Также Пол и Мэйтрэ доказали коррелированность перестановки и ключа. Последняя работа также использует коррелированность ключа и перестановки для того, чтобы создать первый алгоритм полного восстановления ключа из последней перестановки после KSA, не делая предположений о ключе и векторе инициализации(V or Initial Vector). Этот алгоритм имеет постоянную вероятность успеха в зависимости от времени, которая соответствует квадратному корню из сложности полного перебора. Позднее было сделано много работ о восстановлении ключа из внутреннего состояния RC4.

Атака Флурера, Мантина и Шамира (ФМШ)

В 2001 году, Флурер, Мантин и Шамир опубликовали работу об уязвимости ключевого расписания RC4. Они показали, что среди всех возможных ключей, первые несколько байт ключевого потока являются совсем неслучайными. Из этих байт можно с высокой вероятностью получить информацию о используемом шифром ключе. И если долговременный ключ и оказия (nonce) просто конкатенируются для создания ключа шифра RC4, то этот долговременный ключ может быть получен с помощью анализа достаточно большого количества сообщений, зашифрованных с использованием данного ключа. Эта уязвимость и некоторые связанные с ней эффекты были использованы при взломе шифрования WEP в беспроводных сетях стандарта IEEE 802.11. Это показало необходимость скорейшей замены WEP, что повлекло за собой разработку нового стандарта безопасности беспроводных сетей WPA.

Криптосистему можно сделать невосприимчивой к этой атаке, если отбрасывать начало ключевого потока. Таким образом, модифицированный алгоритм называется «RC4-drop[n]», где n — количество байт из начала ключевого потока, которые следует отбросить. Рекомендовано использовать n = 768, консервативная оценка составляет n = 3072.

Атака Кляйна

В 2005 году Андреас Кляйн представил анализ шифра RC4, в котором он указал на сильную коррелированность ключа и ключевого потока RC4. Кляйн проанализировал атаки на первом раунде (подобные атаке ФМШ), на втором раунде и возможные их улучшения. Он также предложил некоторые изменения алгоритма для усиления стойкости шифра. В частности, он утверждает, что если поменять направление цикла на обратное в алгоритме ключевого расписания, то можно сделать шифр более стойким к атакам типа ФМШ.

Комбинаторная проблема

В 2001 году Ади Шамир и Ицхак Мантин первыми поставили комбинаторную проблему, связанную с количеством всевозможных входных и выходных данных шифра RC4. Если из всевозможных 256 элементов внутреннего состояния шифра известно x элементов из состояния ($x \leq 256$), то, если предположить, что остальные элементы нулевые, максимальное количество элементов, которые могут быть получены детерминированным алгоритмом за следующие 256 раундов также равно x. В 2004 году это предположение было доказано Сорадюти Полом (Souradyuti Paul) и Бартом Прэнилом (Bart Preneel).

Программная реализация

Работа многих поточных шифров основана на линейных регистрах сдвига с обратной связью (LFSR). Это позволяет достичь высокой эффективности реализаций шифра в виде ИС. Но затрудняет программную реализацию таких шифров. Поскольку шифр RC4 не использует LFSR и основан на байтовых операциях, его удобно реализовывать программно. Типичная реализация выполняет от 8 до 16 машинных команд на каждый байт текста, поэтому программная реализация шифра должна работать очень быстро.

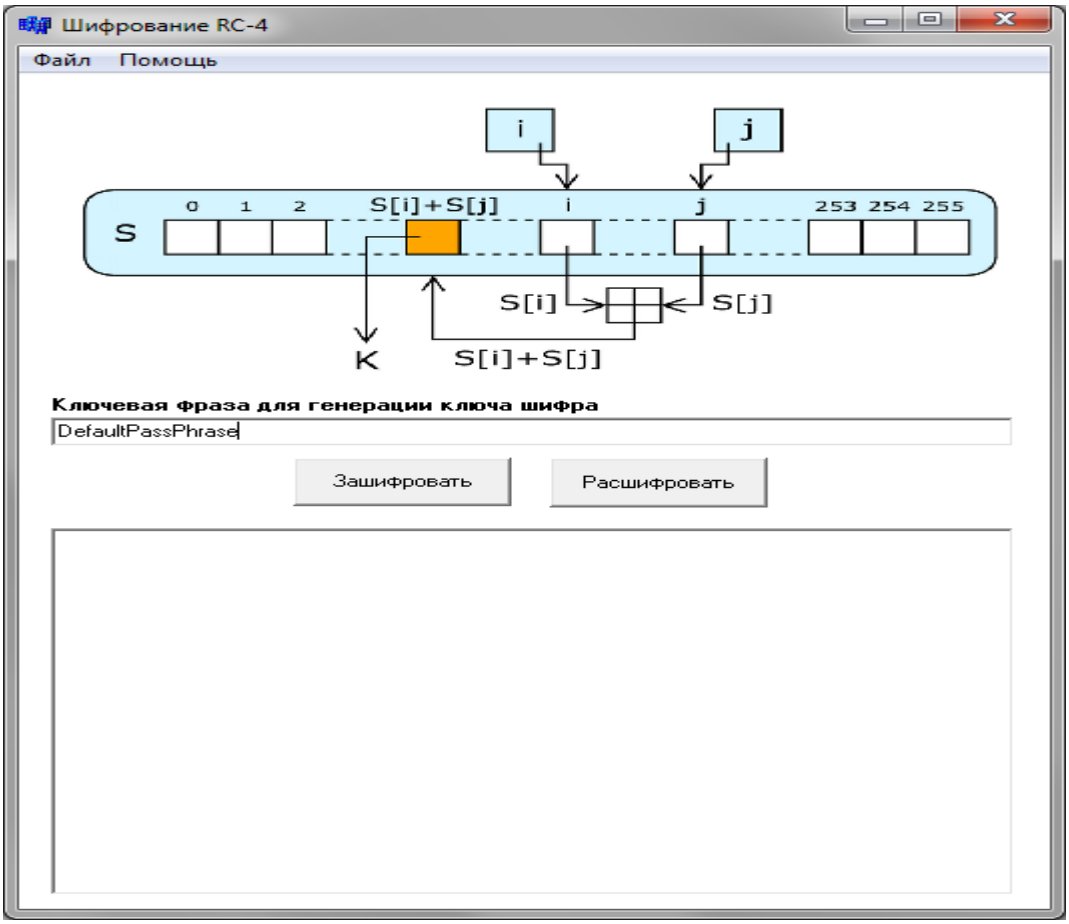


Рис. 5.2. Общий вид программы

Файл

Содержит:

Зашифровать – выбор файла, который необходимо зашифровать. При нажатии открывается интерфейс, позволяющий выбрать файл в проводнике.

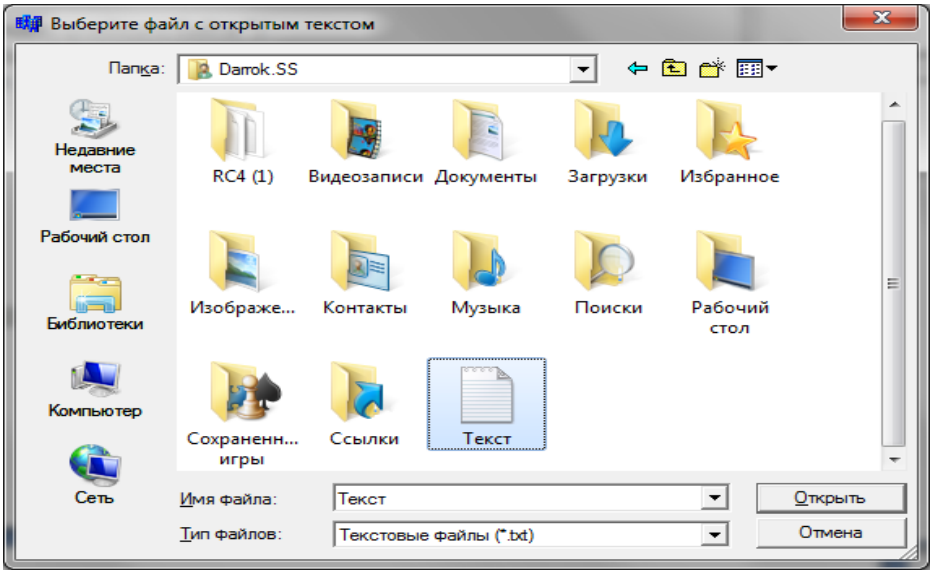


Рис. 5.3. Выбор текстового файла для шифрования

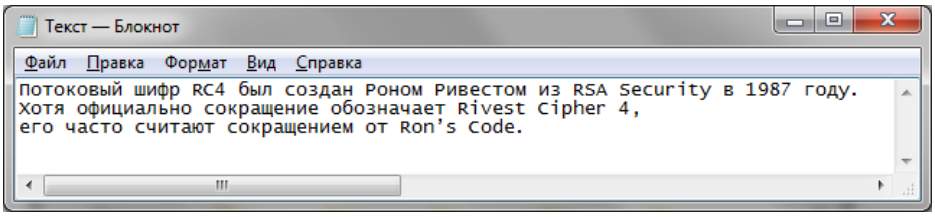


Рис. 5.4. Шифруемый текст

Расшифровать – выбор файла, который необходимо расшифровать. При нажатии открывается интерфейс, позволяющий выбрать файл в проводнике.

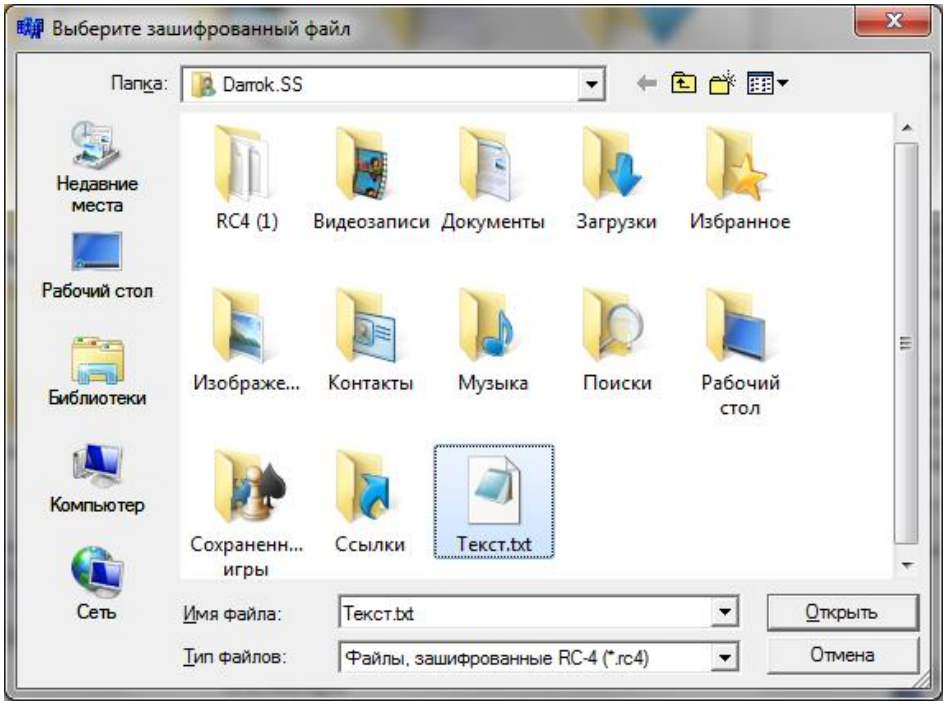


Рис. 5.4. Выбор текстового файла для расшифрования

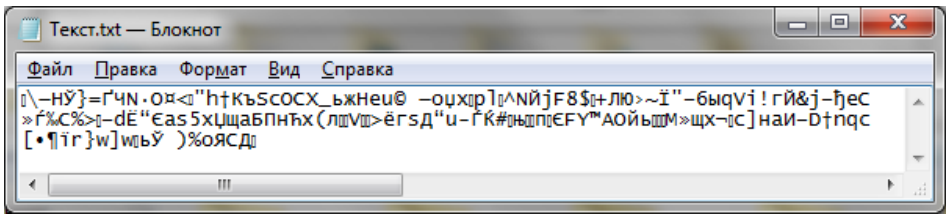


Рис. 5.5. Зашифрованный текст

Выход – выход из программы.

Помощь

Содержит:

Алгоритм – отображает теоретическую информацию об алгоритме, реализуемом в лабораторной работе.

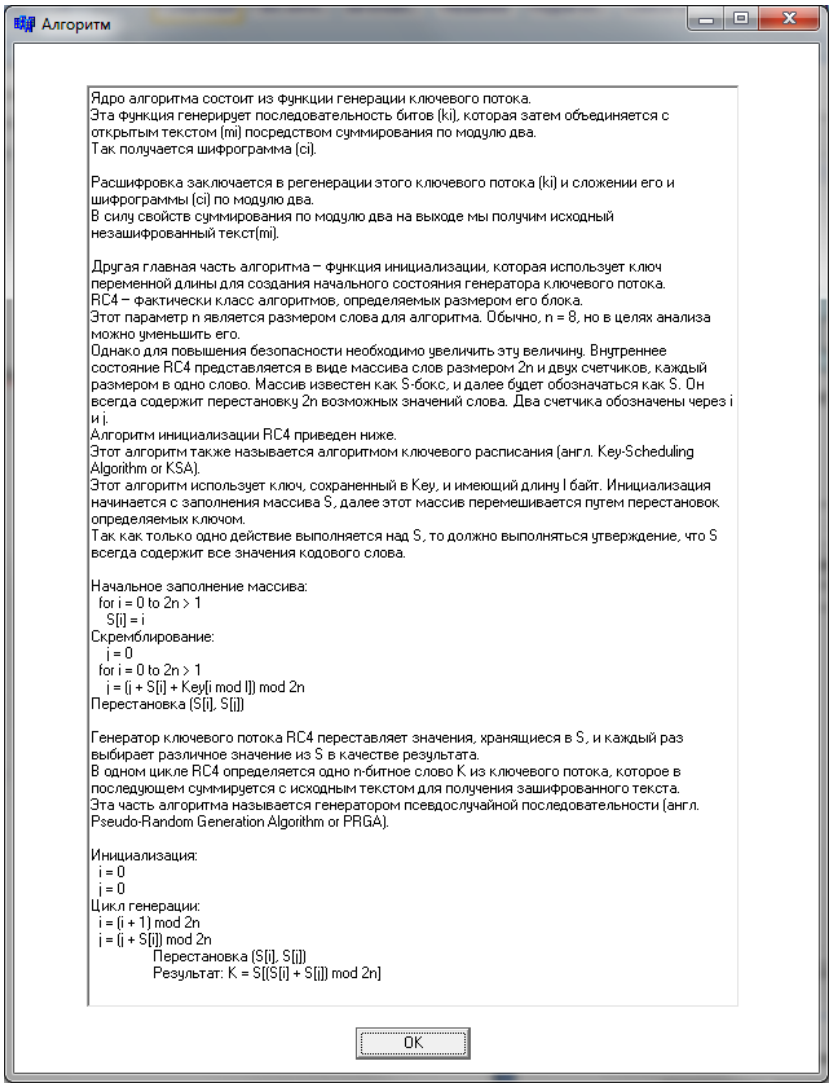


Рис. 5.6. Описание работы алгоритма

Автор – отображает информацию об авторе лабораторной работы и его научного руководителя.

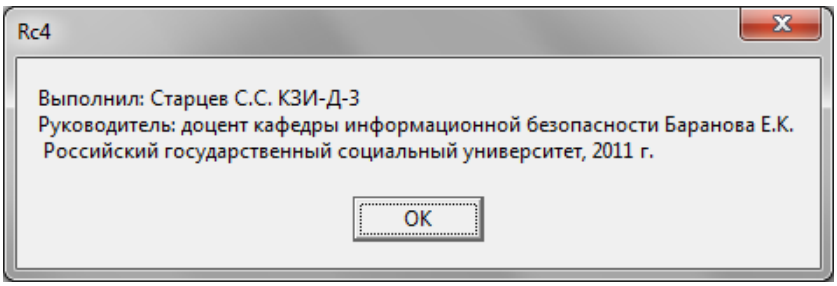


Рис. 5.7. Информация об авторе и руководителе

Поле “Ключ”

В это поле вводятся символы, являющиеся Ключом, который необходим для реализации шифрования методом, используемым в алгоритме.

Окно процесса шифрования

Отображает процесс шифрования и расшифрования, а именно: инициализацию матрицы, выполнение перестановок, считывание блока, количество обработанных байт и запись блока.

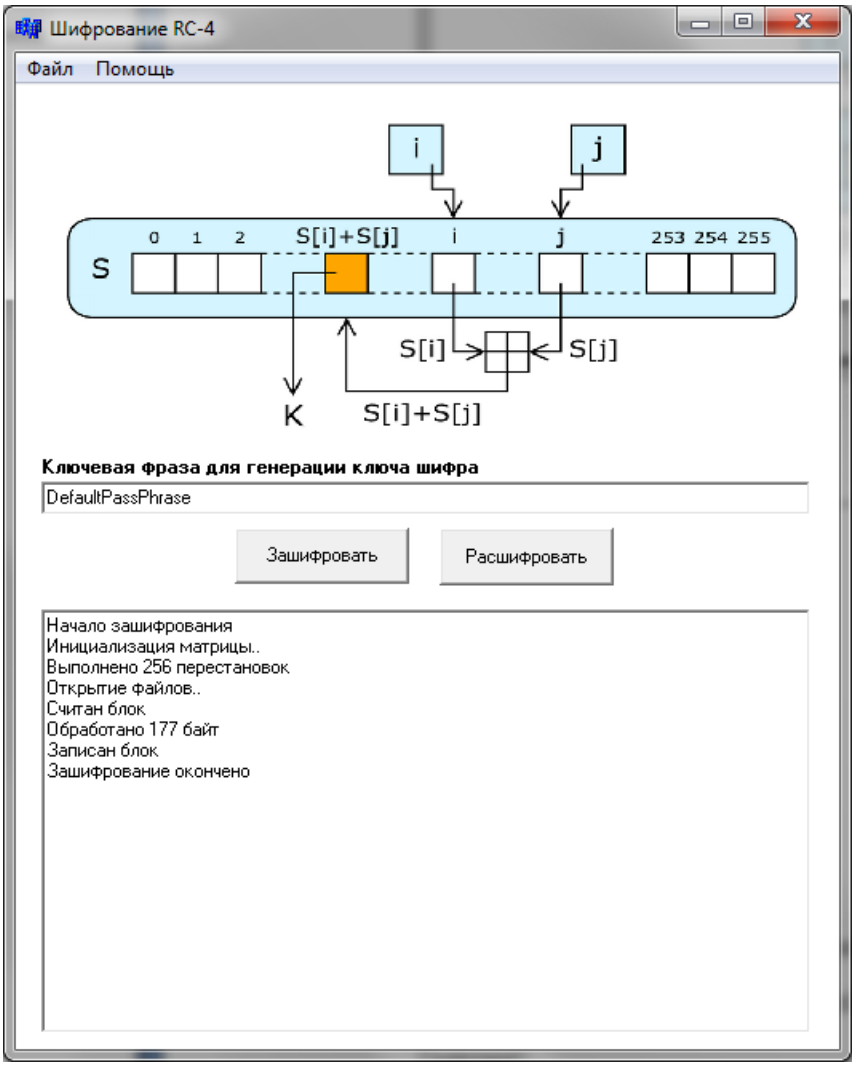


Рис.5.8. Отображение процесса шифрования

Последовательность действий

Выбрать входной и выходный файлы, ввести ключ, нажать «Зашифровать/Расшифровать». Т.к. алгоритм симметричный, то процессы шифрования и расшифрования аналогичны.

Заключение

Была поставлена и, в ходе подготовки данной лабораторной работы, достигнута цель: разработка программы, реализующей алгоритм шифрования RC4, а так же рассмотрены и изучены особенности этого алгоритма.

Потоковый шифр RC4 был разработан Роном Ривестом в 1987 году. Этот шифр позволяет использовать ключи размером от 8 до 2048 бит (с шагом 8). В RC4 для

зашифрования и расшифрования применяются одни и те же действия: генерируется гамма, которая накладывается на шифруемое сообщение путем сложения по модулю 2 (операция XOR).

RC4 применяется в таких продуктах, как Microsoft Office, Lotus Notes, Adobe Acrobat и др.

Алгоритм RC4 является собственностью компании RSA Data Security, Inc. Его описание никогда не было опубликовано и предоставлялось партнерам только после подписания соглашения о неразглашении. Однако в сентябре 1994 года в списке рассылки Cipherpunks (Шифропанки) кто-то анонимно опубликовал алгоритм шифрования, который на всех известных тестовых значениях совпадал с RC4. С тех пор сам алгоритм перестал быть секретом, но название RC4 остается торговой маркой. То есть, чтобы получить право заявлять, что в коммерческом программном продукте используется RC4, необходимо приобрести лицензию на этот алгоритм у RSA Data Security. А без лицензии можно утверждать лишь то, что "используется алгоритм, похожий на RC4 и совпадающий с ним на всем известном множестве тестов". Именно поэтому на языке ADA был реализован Alleged (предполагаемый) RC4.

Лабораторная работа 6. Исследование отечественных стандартов функции хэширования и цифровой подписи

Цель работы

Целью данной лабораторной работы является изучение процессов вычисления функции хэширования по ГОСТ Р 34.11-94, формирования и проверки ЭЦП на эллиптических кривых по ГОСТ Р 34.10-2001.

Краткие теоретические сведения

Электронная Цифровая Подпись

При обмене электронными документами по сети связи существенно снижаются затраты на обработку и хранение документов, убыстряется их поиск. Но при этом возникает проблема аутентификации автора документа и самого документа, т.е. установления подлинности автора и отсутствия изменений в полученном документе. В обычной (бумажной) информатике эти проблемы решаются за счет того, что информация в документе и рукописная подпись автора жестко связаны с физическим носителем (бумагой). В электронных документах на машинных носителях такой связи нет.

Целью аутентификации электронных документов является их защита от возможных видов злоумышленных действий, к которым относятся:

- *активный перехват* – нарушитель, подключившийся к сети, перехватывает документы (файлы) и изменяет их;
- *маскарад* – абонент *C* посылает документ абоненту *B* от имени абонента *A*;
- *ренегатство* – абонент *A* заявляет, что не посылал сообщения абоненту *B*, хотя на самом деле послал;
- *подмена* – абонент *B* изменяет или формирует новый документ и заявляет, что получил его от абонента *A*;
- *повтор* – абонент *C* повторяет ранее переданный документ, который абонент *A* посылал абоненту *B*.

Эти виды злоумышленных действий могут нанести существенный ущерб банковским и коммерческим структурам, государственным предприятиям и организациям, частным лицам, применяющим в своей деятельности компьютерные информационные технологии.

При обработке документов в электронной форме совершенно непригодны традиционные способы установления подлинности по рукописной подписи и оттиску печати на бумажном документе. Принципиально новым решением является электронная цифровая подпись (ЭЦП).

Электронная цифровая подпись предназначена для аутентификации лица, подписавшего электронное сообщение. Функционально она аналогична обычной рукописной подписи.

Электронная цифровая подпись обладает следующими свойствами:

- осуществляет контроль целостности передаваемого подписанного сообщения;
- предоставляет возможность доказательно подтвердить авторство лица, подписавшего сообщение;
- не дает самому этому лицу возможности отказаться от обязательств, связанных с подписанным текстом;
- обеспечивает защиту от возможностей подделки сообщения.

Цифровая подпись представляет собой относительно небольшое количество дополнительной цифровой информации, передаваемой вместе с подписываемым текстом.

Система ЭЦП включает две процедуры:

- процедуру формирования подписи;
- процедуру проверки подписи.

В процедуре постановки подписи используется секретный ключ отправителя сообщения, в процедуре проверки подписи – открытый ключ отправителя.

При формировании ЭЦП отправитель прежде всего вычисляет хэш-функцию h(M) подписываемого текста M. Вычисленное значение хэш-функции h(M) представляет собой один короткий блок информации m, характеризующий весь текст M в целом. Затем число m шифруется секретным ключом отправителя. Получаемая при этом пара чисел представляет собой ЭЦП для данного текста M.

При проверке ЭЦП получатель сообщения снова вычисляет хэш-функцию m = h(M) принятого по каналу текста M, после чего при помощи открытого ключа отправителя проверяет, соответствует ли полученная подпись вычисленному значению m хэш-функции.

Принципиальным моментом в системе ЭЦП является невозможность подделки ЭЦП пользователя без знания его секретного ключа подписывания.

В качестве подписываемого документа может быть использован любой файл.

Функция хэширования

Функция хэширования (хэш-функция, или дайджест-функция) представляет собой отображение, на вход которого подается сообщение переменной длины M, а выходом является строка фиксированной длины h (M). Иначе говоря, хэш-функция h (·) принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение h (M) фиксированной длины.

Хэш-значение h (M) – это сжатое двоичное представление (дайджест) основного сообщения M произвольной длины. Функция хэширования позволяет сжать подписываемый документ M до нескольких десятков или сотен бит (например, 128 или 256 бит), тогда как M может быть размером в мегабайт или более. Следует отметить, что значение хэш-функции h (M) зависит от документа M сложным образом и не позволяет восстановить сам документ M.

Функция хэширования может использоваться для обнаружения модификации сообщения, то есть служить в качестве криптографической контрольной суммы (также называемой кодом обнаружения изменений или кодом аутентификации сообщения). В этом качестве функция хэширования применяется при формировании и проверке электронной цифровой подписи.

Модулярная арифметика

Пусть a и n – натуральные числа. «Разделить число a на число n с остатком» – это значит найти целые числа q и r, удовлетворяющие условию

$$a = q \cdot n + r, \; 0 \le r < n.$$

При этом число q называют неполным частным, а r – остатком от деления числа a на число n.

Если остаток r равен нулю, то говорят, что число n делит число a, или, по-другому, n является делителем числа a.

Целые числа a и b называются *сравнимыми по модулю* n, если их остатки при делении на n совпадают. Обычно для обозначения этого факта используется запись

$$a \equiv b \pmod{n}.$$

Отсюда, в частности, следует, что число n делит разность чисел a и b. Для обозначения остатка обычно используют бесскобочную запись

$$b = a \bmod n.$$

Операцию нахождения числа $b = a \bmod n$ называют *приведением числа a по модулю n.*

Конечное множество целых чисел $a_0, \ldots, a_{n-1}$, таких, что для любого целого числа b найдется $k \in \{0, \ldots, n-1\}$ со свойством $a_k \equiv b \pmod{n}$, называется *полной системой вычетов по модулю n.* Обычно используется полная система вычетов $\{0, \ldots, n-1\}$.

Эллиптические кривые

Эллиптические кривые применяются в криптографии с 1985 года. Интерес к ним обусловлен тем, что на их основе обеспечиваются те же криптографические свойства, которыми обладают числовые или полиномиальные криптосистемы, но при существенно меньшем размере ключа.

Пусть задано простое число $p > 3$. Тогда *эллиптической кривой E*, определённой над конечным простым полем F_p, называется кривая, которая задается уравнением вида

$$y^2 \equiv x^3 + ax + b \pmod{p},$$

где $a, b \in F_p$, и $4a^3 + 27b^2$ не сравнимо с нулем по модулю p.

Эллиптическая кривая E состоит из всех точек (x, y), $x, y \in F_p$, удовлетворяющих тождеству (1.4), и бесконечно удаленной точки O, которая называется *нулевой точкой*.

Точки эллиптической кривой будем обозначать $Q = (x, y)$ или просто Q. Две точки эллиптической кривой равны, если равны их соответствующие x- и y-координаты.

Инвариантом эллиптической кривой называется величина $J(E)$, удовлетворяющая тождеству

$$J(E) \equiv 1728 \cdot \frac{4a^3}{4a^3 + 27b^2} \pmod{p}$$

Коэффициенты a, b эллиптической кривой E, по известному инварианту $J(E)$, определяются следующим образом

$$\begin{cases} a \equiv 3k \pmod{p}, \\ b \equiv 2k \pmod{p}, \end{cases} \quad \text{где } k \equiv \frac{J(E)}{1728 - J(E)} \pmod{p},\ J(E) \neq 0 \text{ или } 1728.$$

Таким образом эллиптическая кривая E может быть однозначно задана либо своим инвариантом $J(E)$, либо коэффициентами $a, b \in F_p$.

Функция хэширования ГОСТ Р 34.11-94

Российский стандарт ГОСТ Р 34.11-94 определяет алгоритм и процедуру вычисления хэш-функции для любых последовательностей двоичных символов, применяемых в криптографических методах обработки и защиты информации. Этот стандарт базируется на блочном алгоритме шифрования ГОСТ 28147-89, хотя в принципе можно было бы использовать и другой блочный алгоритм шифрования с 64-битовым блоком и 256-битовым ключом.

Введем обозначения, которые будут использоваться в данном пункте:

$\{01\}^*$ – множество двоичных строк произвольной конечной длины;

$\{01\}^n$ – множество двоичных строк длиной n бит;

$\{0\}^n$ – двоичная строка из n нулей;

$|A|$ – длина слова A в битах;

$A \oplus B$ – побитное сложение слов A и B по mod 2, или попросту *XOR*;

$A \otimes B$ –сложение слов A и B по mod 2^k, где $k = |A| = |B|$;

$A \boxplus B$ – сложение слов A и B по mod 2^{256};

$\leftarrow$ оператор присвоения;

|| – конкатенация.

Сообщения с произвольной длины можно сжать используя хэш-функцию с фиксированным размером входа при помощи двух методов:

- последовательного (итерационного);
- параллельного.

Создатели ГОСТ Р 34.11-94 пошли по первому пути и использовали метод последовательного хэширования использующий хэш-функцию с фиксированным размером входа h: $\{01\}^{2n} \to \{01\}^{n}$ (см. рисунок 1.1), то есть функцию сжатия с коэффициентом 2.

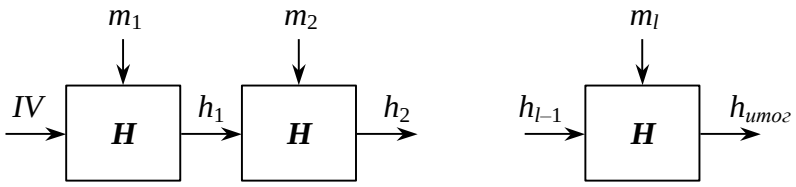


Рис. 6.1. Метод последовательного хэширования

Если необходимо хэшировать сообщение $m = (m_1, m_2, \ldots, m_l)$, то хэширование выполняется следующим образом:

$h_0 \leftarrow IV$,

$h_i \leftarrow H(m_i, h_{i-1})$, для $i = 1, 2, \ldots, l$,

$h_{итог} \leftarrow h_l$.

Здесь H – функция сжатия, а h_i – переменная сцепления, IV – начальный вектор.

Если последний блок меньше чем n бит, то он набивается одним из существующих методов до достижения длины кратной n. В отличие от стандартных предпосылок, что сообщение разбито на блоки и произведена набивка последнего блока (форматирование входа априори), если необходимо, до начала хэширования, то в ГОСТ Р 34.11-94 процедура хэширования ожидает конца сообщения (форматирование входного сообщения постериори). Набивка производится следующим образом: последний блок сдвигается вправо, а затем набивается нулями до достижения длины в 256 бит (рисунок 6.2).

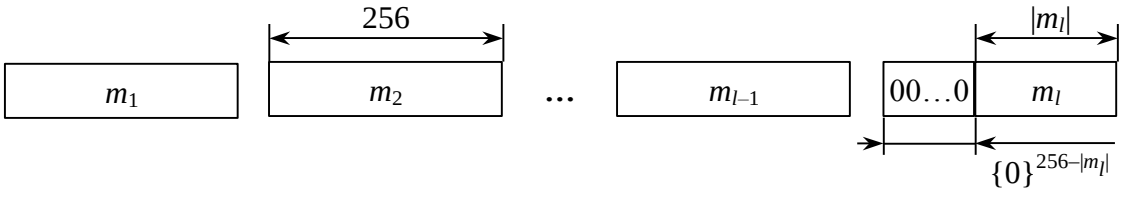


Рис. 6.2. Набивка сообщения

Указывать в передаваемом сообщении сколько было добавлено нулей к последнему блоку не требуется, так как длина сообщения участвует в хэшировании.

Параллельно рассчитываются контрольная сумма, представляющая собой сумму всех блоков сообщения (последний суммируется уже набитым) по правилу $A \otimes B$, и битовая длина хэшируемого сообщения, приводимая по mod 2^{256}, которые в финальной функции сжатия используются для вычисления итогового хэша (рисунок 6.3).

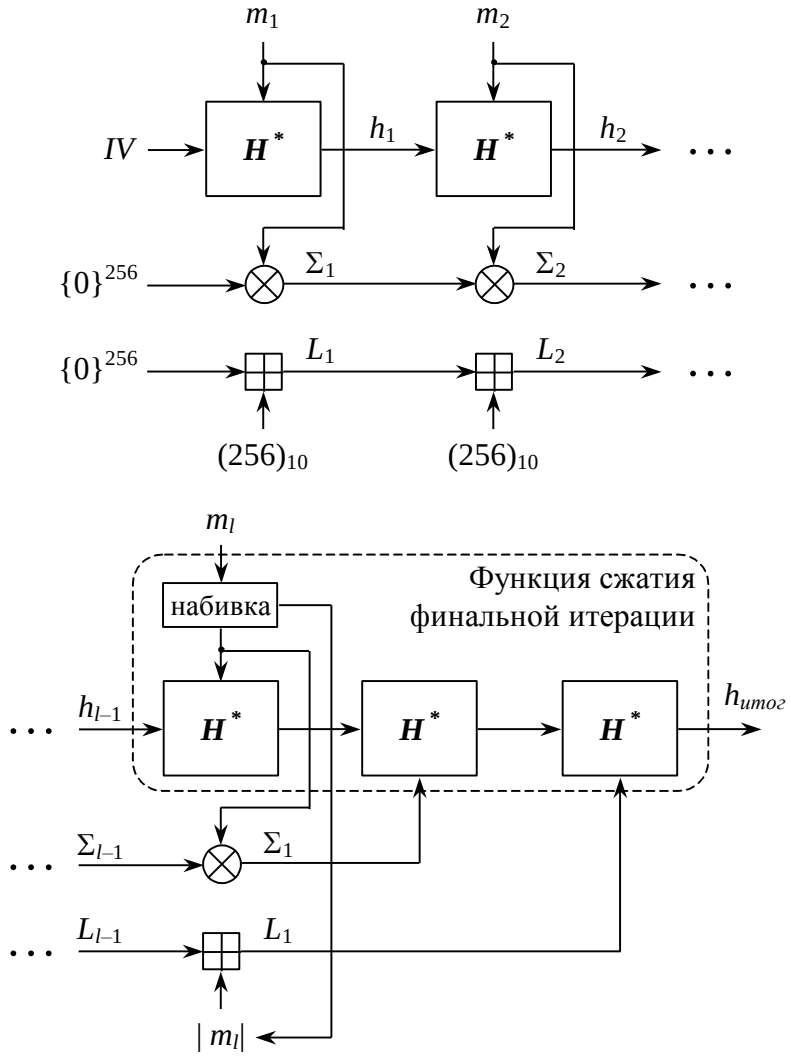


Рис. 6.3. Общая схема функции хэширования по ГОСТ Р 34.11-94

Согласно ГОСТ Р 34.11-94 IV – произвольное фиксированное слово длиной 256 бит ($IV \in \{01\}^{256}$). В таком случае, если он априорно не известен верифицирующему целостность сообщения, то он должен передаваться вместе с сообщением с гарантией целостности. При небольших сообщениях можно усложнить задачу противнику, если IV выбирается из небольшого множества допустимых величин (но при этом увеличивается вероятность угадывания хэш величины противником). Также он может задаваться в рамках организации, домена как константа (обычно как в тестовом примере).

Функция сжатия внутренних итераций

Функция сжатия внутренних итераций (по ГОСТ “шаговая функция хэширования”) H^* отображает два слова длиной 256 бит в одно слово длиной 256 бит:

H^*: $\{01\}^{256} \times \{01\}^{256} \rightarrow \{01\}^{256}$,

и состоит из трех процедур (рисунок 3.5):

- перемешивающего преобразования;
- шифрующего преобразования;
- генерирования ключей.

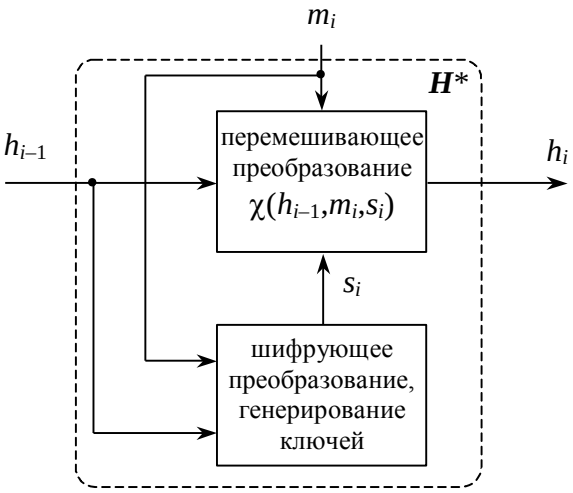


Рис. 6.4. Структура функции сжатия в ГОСТ Р 34.11-94

Генерирования ключей

Данная подпрограмма использует следующие функции и константы:

- Константы

$C_2 = C_4 = \{0\}^{256}$ и $C_3 = 1^8 0^8 1^{16} 0^{24} 1^{16} 0^8 (0^8 1^8)^2 1^8 0^8 (0^8 1^8)^4 (1^8 0^8)^4$,

где степень обозначает количество повторений 0 или 1.

- Функции

- P: $\{01\}^{256} \rightarrow \{01\}^{256}$.

Пусть $X = \xi_{32} \| \xi_{31} \| \ldots \| \xi_1$,

тогда $P(X) = \xi_{\varphi(32)} \| \xi_{\varphi(31)} \| \ldots \| \xi_{\varphi(1)}$, где $\varphi(i+1+4(k-1)) = 8i+k$, $i = 0\ldots3$, $k = 1\ldots8$.

- A: $\{01\}^{256} \rightarrow \{01\}^{256}$.

Пусть $X = x_4 \| x_3 \| x_2 \| x_1$,

тогда $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2$.

При вычислении ключей реализуется следующий алгоритм (рисунок 1.5):

1. $j \leftarrow 1$, $V \leftarrow m_i$, $U \leftarrow h_{i-1}$;
2. $W \leftarrow U \oplus V$, $K_1 \leftarrow P(W)$;
3. $j \leftarrow j + 1$;

4. Проверить условие $j = 5$. При положительном исходе перейти к шагу 7, при отрицательном – к шагу 5;

5. $U \leftarrow A(U) \oplus C_j$, $V \leftarrow A(A(V))$, $W \leftarrow U \oplus V$, $K_j \leftarrow P(W)$;

6. Перейти к шагу 3;

7. Выход (K_1, K_2, K_3, K_4).

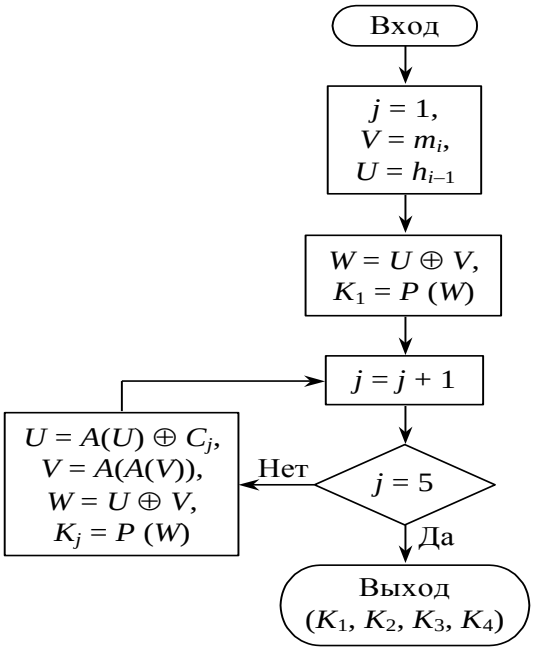


Рис. 6.5. Алгоритм генерирования ключей

Шифрующее преобразование

Основным функциональным предназначением является получение s_i из h_{i-1}.

Пусть $h_{i-1} = h_{i-1}^4 \parallel h_{i-1}^3 \parallel h_{i-1}^2 \parallel h_{i-1}^1$, где $h_{i-1}^j \in \{01\}^{64}$, $j = 1\ldots4$, а

$s_i = s_i^4 \parallel s_i^3 \parallel s_i^2 \parallel s_i^1$, где $s_i^j \in \{01\}^{64}$, $j = 1\ldots4$.

Тогда

$s_i^j \leftarrow E_{K_j}(h_{i-1}^j)$,

где $j = 1\ldots4$, E_{K_j} – шифрование по ГОСТ 28147-89 в режиме простой замены.

Схематически это изображено на рисунке 1.6.

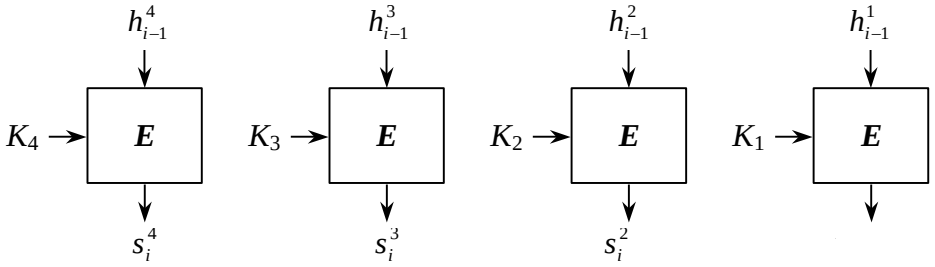


Рис. 6.6. Шифрующее преобразование

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,
- и один дополнительный режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается массив, подвергаемый криптографическому преобразованию, именно поэтому ГОСТ относится к блочным шифрам.

Если внимательно изучить оригинал ГОСТа 28147-89, можно заметить, что в нем содержится описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня, называемые в тексте ГОСТа *циклами*. Эти фундаментальные алгоритмы упоминаются в данной работе как *базовые циклы*, чтобы отличать их от всех прочих циклов. Они имеют следующие названия и обозначения:

- $Ц_{32\text{-З}}$ – цикл зашифрования (32-З);
- $Ц_{32\text{-Р}}$ – цикл расшифрования (32-Р);

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной единственной процедуры, называемой для определенности далее в настоящей работе *основным шагом криптопреобразования*.

В соответствии с принципом Кирхгофа, которому удовлетворяют все современные известные широкой общественности шифры, секретность зашифрованного сообщения обеспечивается секретностью ключевой информации. В ГОСТе ключевая информация состоит из двух структур данных. Помимо собственно *ключа*, необходимого для всех шифров, она содержит еще и *таблицу замен*:

1. **Ключ** является массивом из восьми 32-битных элементов кода: $K = \{K_i\}_{0 \le i \le 7}$. В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака: $0 \le K_i < 2^{32}$. Таким образом, размер ключа составляет 32·8=256 бит или 32 байта.

2. **Таблица замен** является матрицей 8×16, содержащей 4-битовые элементы, которые можно представить в виде целых чисел от 0 до 15: $H = \{H_{i,j}\}_{0 \le j \le 15}^{0 \le i \le 7}$, $0 \le H_{i,j} \le 15$. Строки *таблицы замен* называются *узлами замен*, они должны содержать различные значения, то есть каждый узел замен должен содержать 16 различных чисел от 0 до 15 в произвольном порядке. Таким образом, общий объем таблицы замен равен: 8 узлов × 16 элементов/узел × 4 бита/элемент = 512 бит или 64 байта.

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена на рисунке 1.7.

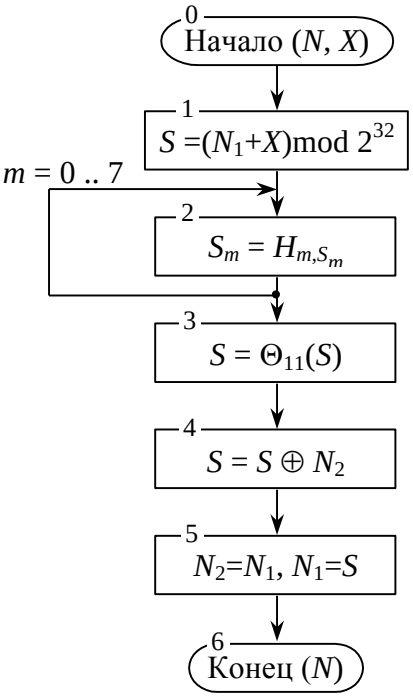


Рис. 6.7. Схема основного шага криптопреобразования
алгоритма ГОСТ 28147-89

Ниже даны пояснения к алгоритму основного шага:

0. Определяет исходные данные для основного шага криптопреобразования:

- N – преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать $N = (N_1, N_2)$.
- X – 32-битовый элемент ключа.

1. Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий шаг.

2. Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода:

$S = (S_7, S_6, S_5, S_4, S_3, S_2, S_1, S_0)$.

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен следующим образом: значение блока S_i заменяется на S_i-тый по порядку элемент (нумерация с нуля) i-того узла замен (т.е. i-той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент

из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битных элементов в 32-битном блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битного блока данных, равному как известно 2^4, шестнадцати.

3. Циклический сдвиг на 11 бит влево. Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом Θ_{11} обозначена функция циклического сдвига своего аргумента на 11 бит в сторону старших разрядов.

4. Побитовое сложение: значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

5. Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

6. Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

Базовые циклы заключаются в многократном выполнении *основного шага* с использованием разных элементов ключа и отличаются друг от друга только числом повторения шага и порядком использования ключевых элементов. Ниже приведен этот порядок для различных циклов.

1. Цикл зашифрования 32-З:

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$.

2. Цикл расшифрования 32-Р:

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$.

Каждый из циклов имеет собственное буквенно-цифровое обозначение, где первый элемент обозначения, задает число повторений основного шага в цикле, а второй элемент обозначения, буква, задает порядок зашифрования («З») или расшифрования («Р») в использовании ключевых элементов.

Цикл расшифрования должен быть обратным циклу зашифрования, то есть последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок, что отражается следующим соотношением:

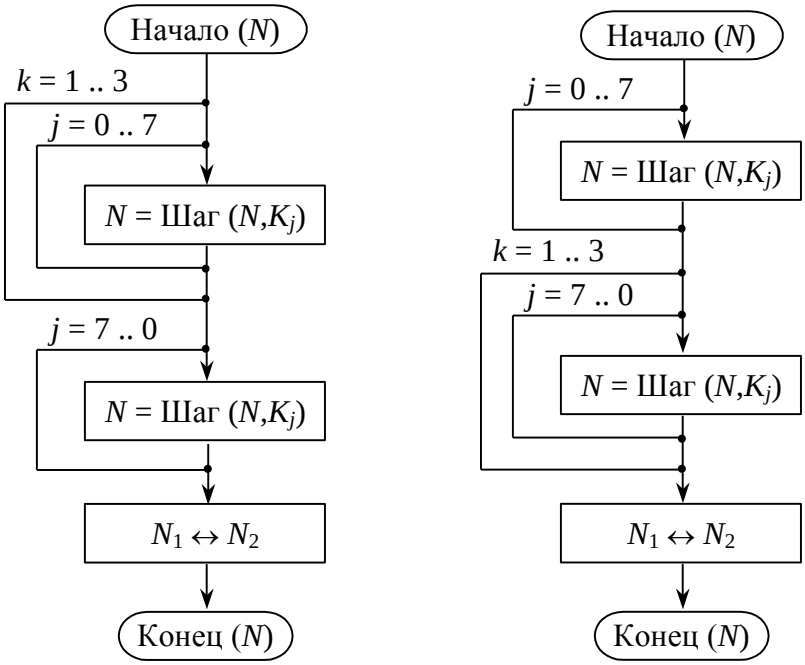
$Ц_{32\text{-}Р}(Ц_{32\text{-}З}(T))=T$,

где T – произвольный 64-битный блок данных,

$Ц_X(T)$ – результат выполнения цикла X над блоком данных T.

Для выполнения этого условия для алгоритмов, подобных ГОСТу, необходимо и достаточно, чтобы порядок использования ключевых элементов соответствующими циклами был взаимно обратным. Из двух взаимно обратных циклов любой может быть использован для зашифрования, тогда второй должен быть использован для расшифрования данных, однако стандарт ГОСТ28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе.

Схемы базовых циклов приведены на рисунках 1.8а, 1.8б. Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битный блок данных, обозначенный на схемах *N*.



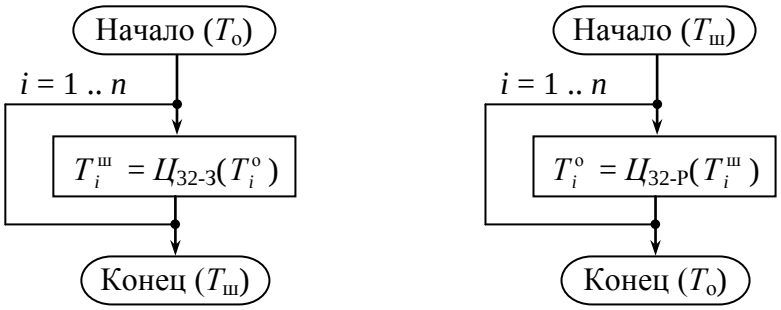
а) б)

Рис. 6.8. Схемы цикла зашифрования 32-З (а) и расшифрования 32-Р (б)

Символ Шаг(*N*,*X*) обозначает выполнение основного шага криптопреобразования для блока *N* с использованием ключевого элемента *X*.

В конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для их взаимной обратимости.

Как уже упоминалось выше, нас интересует шифрование в режиме простой замены. Зашифрование в режиме простой замены заключается в применении цикла 32-З к блокам открытых данных, расшифрование – цикла 32-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены на рисунках 3.10а и 3.10б соответственно.



а) б)

Рис. 6.9. Алгоритмы зашифрования (а) и расшифрования (б)

данных в режиме простой замены

На рисунке 3.10 используются следующие обозначения:

- $T_о$, $T_ш$ – массивы соответственно открытых и зашифрованных данных;
- $T_i^о$, $T_i^ш$ – i-тые по порядку 64-битные блоки соответственно открытых и зашифрованных данных: $T_о = (T_n^о, \ldots, T_2^о, T_1^о)$, $T_ш = (T_n^ш, \ldots, T_2^ш, T_1^ш)$, $1 \le i \le n$;
- n – число 64-битных блоков в массиве данных.

Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам: $|T_о| = |T_ш| = 64 \cdot n$, после выполнения операции размер полученного массива данных не изменяется.

Перемешивающее преобразование

Перемешивающее преобразование имеет вид

$h_i = \chi(m_i, h_{i-1}, s_i) = \psi^{61}(h_{i-1} \oplus \psi(m_i \oplus \psi^{12}(s_i)))$,

где ψ^j – j-я степень преобразования ψ.

Схематически данное преобразование представлено на рисунке 1.10.

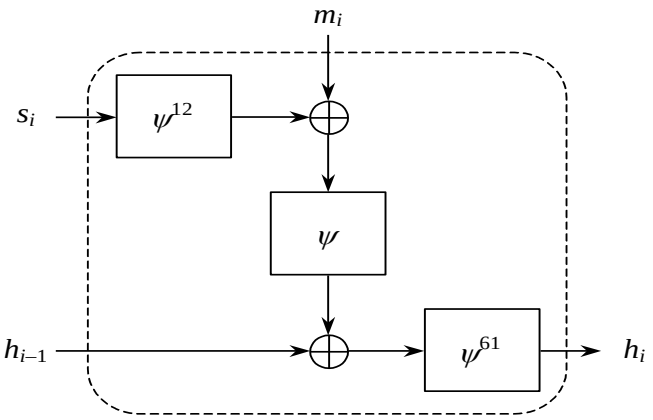


Рис. 6.10. Перемешивающее преобразование ГОСТ Р 34.11-94

Заметим, что s_i выводится из h_{i-1} (см. рисунок 6.11).

Функция ψ: $\{01\}^{256} \rightarrow \{01\}^{256}$ преобразует слово $\eta_{16} \| \eta_{15} \| \ldots \| \eta_1 \in \{01\}^{16}$, i =1…16 в слово $\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16} \| \eta_{16} \| \eta_{15} \| \ldots \| \eta_2$ (рисунок 6.11).

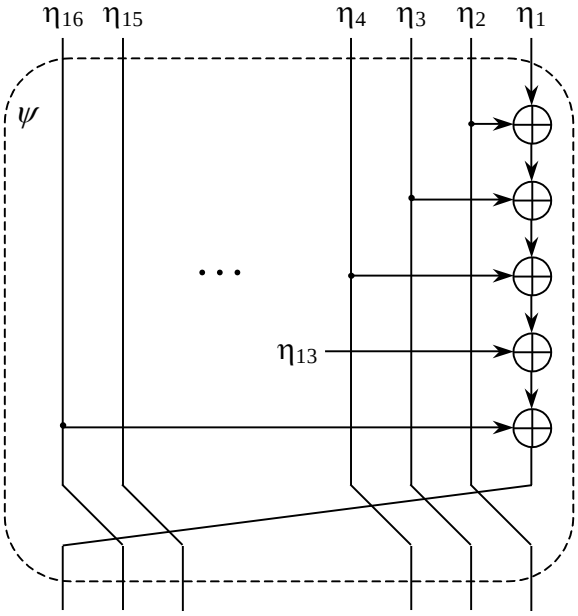


Рис. 6.11. Функция ψ перемешивающего преобразования

Функция сжатия финальной итерации

Функция сжатия финальной итерации производит итоговую хэш величину, зависящую от результата хэширования последовательным методом, контрольной суммы по mod 2 и длины сообщения в битах приведенной по mod 2^{256}:

$$m_l^{/} \times h_{l-1} \times \sum_l \times L_l \rightarrow h_{итог},$$

где $m_l^{/}$, h_{l-1}, $\sum_l$, $L_l \in \{01\}^{256}$, $m_l^{/}$ – последний набитый нулями блок (если необходимо, см. рисунок 1.2).

Сначала вычисляется битовая длина последнего (ненабитого) блока сообщения $|m_l| \leq 256$ бит. Если $|m_l| < 256$, то производится его набивка справа нулями до достижения длины 256 бит и получается новый блок $m_l^{/} \leftarrow \{0\}^{256-|m_l|} \| m_l$. Вычисляются итоговая контрольная сумма $\sum_l \leftarrow \sum_{l-1} \otimes m_l^{/}$ и длина всего сообщения $L_l \leftarrow L_{l-1} + |m_l| \pmod{2^{256}}$. Параллельно выполняется последняя итерация $h_l = H^*(m^{/}, h_{l-1})$. Затем вычисляются значения $h_{l+1} \leftarrow H^*(\sum_l, h_l)$ и $h_{итог} \leftarrow H^*(L_l, h_{l+1})$, давая в результате итоговую хэш величину $h_{итог}$.

Теперь дадим формальное описание алгоритма:

Вход: двоичное сообщение M, блоки замен для шифрования в режиме простой замены по ГОСТ 28147-89, начальный вектор $IV \in \{01\}^{256}$.

Выход: хэш величина $h_{итог}$ для сообщения M.

1. *Инициализация алгоритма*: $h \leftarrow IV$, $\sum \leftarrow \{0\}^{256}$, $L \leftarrow \{0\}^{256}$.

2. *Функция сжатия внутренних итераций*: Пока |M| > 256 выполняем следующее:

2.1. $h \leftarrow H^*(m_s, h)$ (итерация метода последовательного хэширования);

2.2. $L \leftarrow L + 256 \pmod{2^{256}}$ (итерация вычисления длины сообщения);

2.3. $\sum \leftarrow \sum \otimes m_s$ (итерация вычисления контрольной суммы).

3. Иначе (*функция сжатия финальной итерации*)

3.1. $L \leftarrow L + |m| \pmod{2^{256}}$ (вычисление полной длины сообщения);

3.2. $m' \leftarrow \{0\}^{256-|m|} \| m$ (набивка последнего блока);

3.3. $\sum \leftarrow \sum \otimes m'$ (вычисление контрольной суммы сообщения);

3.4. $h \leftarrow H^*(m', h)$;

3.5. $h \leftarrow H^*(\sum, h)$;

3.6. $h_{итог} \leftarrow H^*(L, h)$.

4. *Выход* ($h_{итог}$).

Российский стандарт ЭЦП ГОСТ Р 34.10-2001

Новый отечественный стандарт цифровой подписи обозначается как ГОСТ Р 34.10-2001. В нем используются следующие параметры:

- простое число p – модуль эллиптической кривой, удовлетворяющее неравенству $p>2^{255}$. Верхняя граница данного числа должна определяться при конкретной реализации схемы цифровой подписи;
- эллиптическая кривая E, задаваемая своим инвариантом $J(E)$ или коэффициентами $a, b \in F_p$;
- целое число m – порядок группы точек эллиптической кривой E;
- простое число q – порядок циклической подгруппы группы точек эллиптической кривой E, для которого выполнены следующие условия:

$$\begin{cases} m = nq, \quad n \in Z, \quad n \geq 1, \\ 2^{254} < q < 2^{256}; \end{cases}$$

- точка $P \neq O$ эллиптической кривой E, с координатами (x_p, y_p), удовлетворяющая равенству $qP = O$.
- хэш-функция, отображающая сообщения, представленные в виде двоичных векторов произвольной конечной длины, в двоичные вектора длины 256 бит. Хэш-функция определена в ГОСТ Р 34.11, ее алгоритм будет рассмотрен ниже.

Каждый пользователь схемы цифровой подписи должен обладать личными ключами:

- ключом подписи – целым числом d, удовлетворяющим неравенству $0 < d < q$;

- ключом проверки – точкой эллиптической кривой Q с координатами (x_q, y_q), удовлетворяющей равенству $dP = Q$.

На приведенные выше параметры схемы цифровой подписи накладываются следующие требования:

- должно быть выполнено условие $p^t \neq 1 \pmod q$, для всех целых $t = 1, 2, \ldots B$, где B удовлетворяет неравенству $B \geq 31$;
- должно быть выполнено неравенство $m \neq p$;
- инвариант кривой должен удовлетворять условию $J(E) \neq 0$ или 1728.

Для получения цифровой подписи под сообщением M необходимо выполнить следующие действия (рис. 6.12а):

1. вычислить хэш-значение сообщения M: $\bar{h} = h(M)$;
2. вычислить целое число α, двоичным представлением которого является вектор $\bar{h}$, и определить $e \equiv \alpha \pmod q$. Если $e = 0$, то определить $e = 1$;
3. сгенерировать случайное (псевдослучайное) целое число k, удовлетворяющее неравенству $0 < k < q$;
4. вычислить точку эллиптической кривой $C = kP$ и определить $r \equiv x_c \pmod q$, где x_c - x-координата точки C. Если $r = 0$, то вернуться к шагу 3;
5. вычислить значение $s \equiv (rd + ke) \pmod q$. Если $s = 0$, то вернуться к шагу 3;

Подпись для сообщения M составит пара чисел (r,s).

Для проверки цифровой подписи ζ под полученным сообщением M необходимо выполнить следующие действия (рис. 6.12б):

1. Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись неверна;
2. вычислить хэш-значение полученного сообщения M: $\bar{h} = h(M)$;
3. вычислить целое число α, двоичным представлением которого является вектор $\bar{h}$ и определить $e \equiv \alpha \pmod q$. Если $e = 0$, то определить $e = 1$;
4. вычислить значение $v = e^{-1} \pmod q$;
5. вычислить значения $z_1 \equiv sv \pmod q$, $z_2 \equiv -rv \pmod q$;
6. вычислить точку эллиптической кривой $C = z_1P + z_2Q$ и определить $R \equiv x_c \pmod q$, где x_c – x-координата точки C;
7. если выполнено равенство $R = r$, то подпись принимается, в противном случае, подпись неверна.

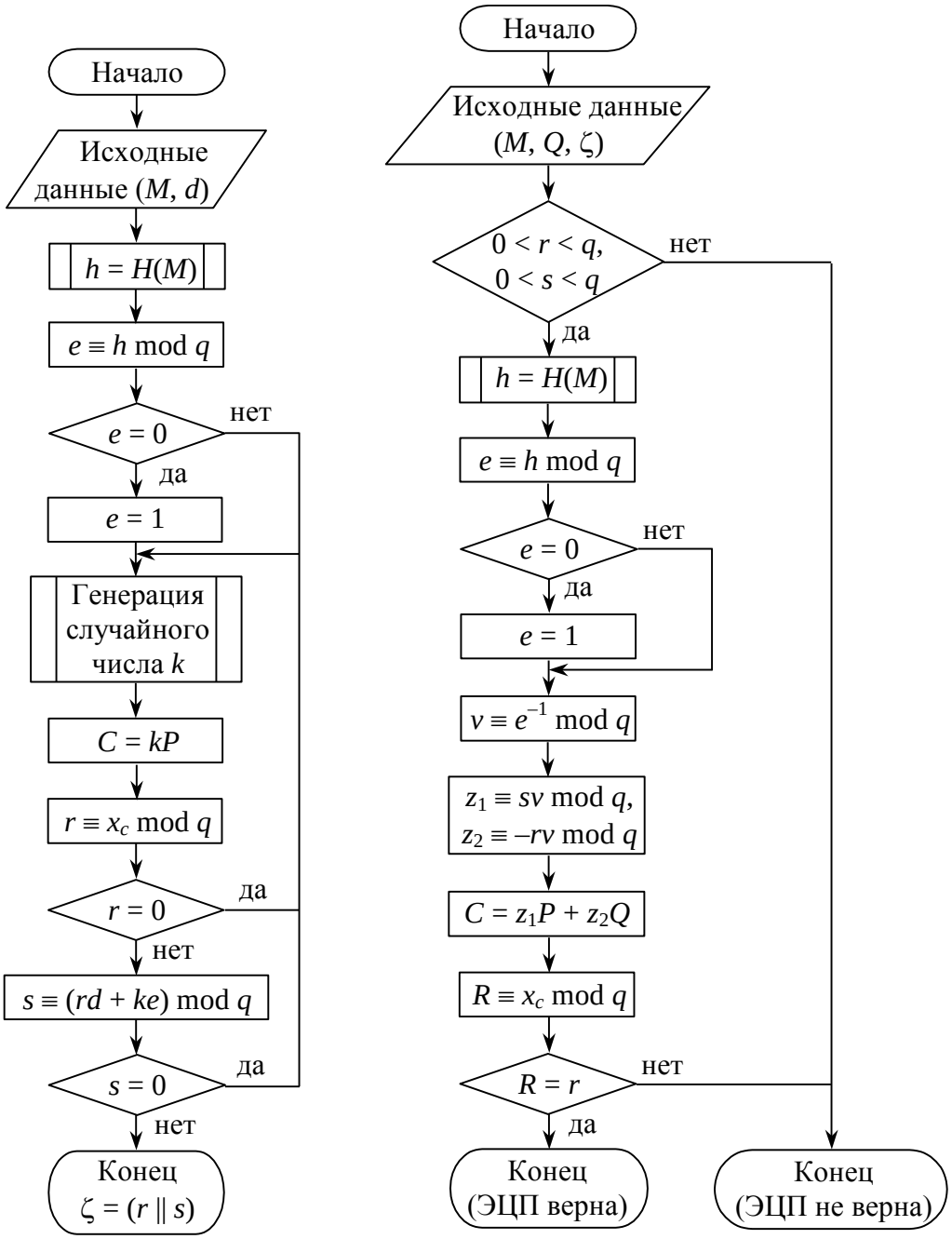


Рис. 6.12. Процесс формирования цифровой подписи (а), процесс проверки цифровой подписи (б).

Интерфейс учебного программного комплекса DigitSign

На рисунке 6.13 приведен интерфейс главного окна учебного программного комплекса.

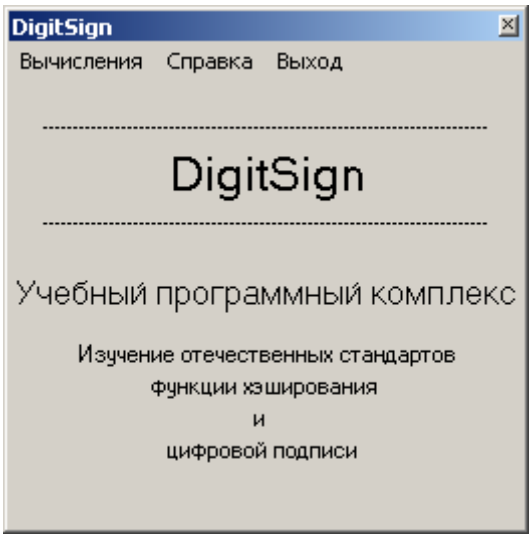


Рис. 6.13. Интерфейс главного окна учебного программного комплекса

Пункт меню “Вычисления” содержит следующие опции:

- Вычисление хэш-функции.

При выборе данной опции открывается окно “Вычисление хэш-функции” (рисунок 6.15), которое позволяет определить хэш-значение произвольного файла при заданных пользователем начальном хэш-векторе и таблице замен. Полученный результат может быть сохранен в файл. Также пользователь получает информацию о времени хэширования сообщения.

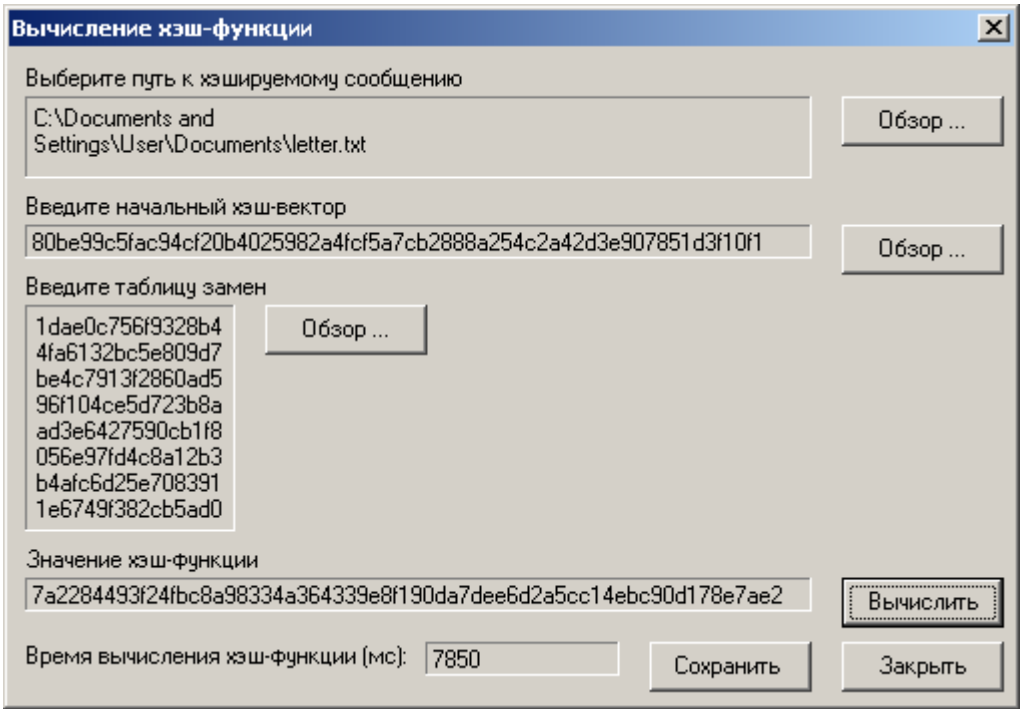


Рис. 6.14. Окно “Вычисление хэш-функции”

- Генератор псевдослучайных чисел.

При выборе данной опции открывается окно “Генератор псевдослучайных чисел” (рисунок 6.16).

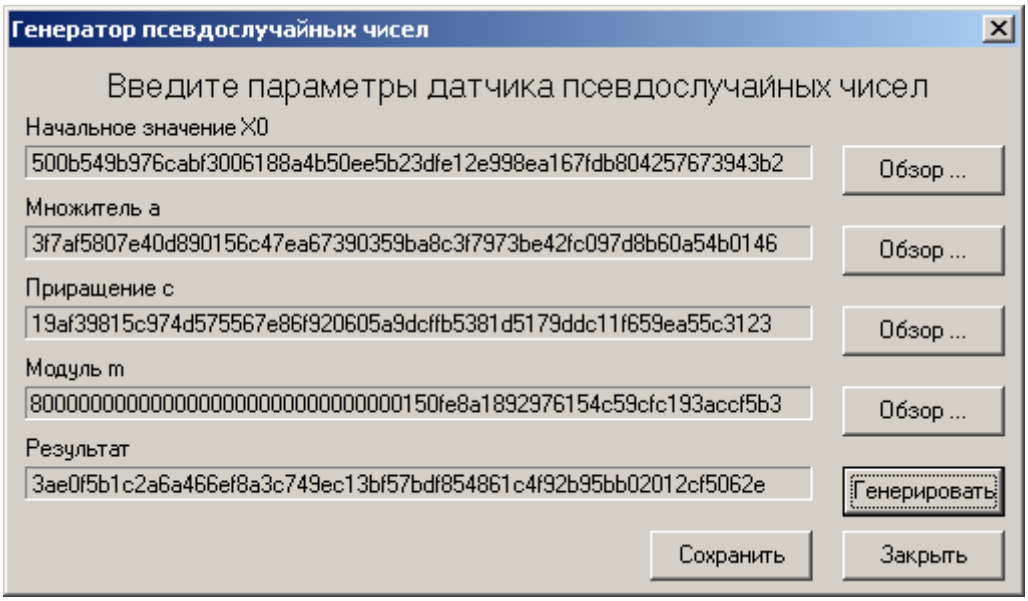


Рис. 6.15. Окно “Генератор псевдослучайных чисел”

Пользователь имеет возможность ввести параметры своего генератора, произвести вычисления и сохранить полученный результат в файл.

- Вычисление открытого ключа.

При выборе данной опции открывается окно “Вычисление открытого ключа” (рисунок 6.16).

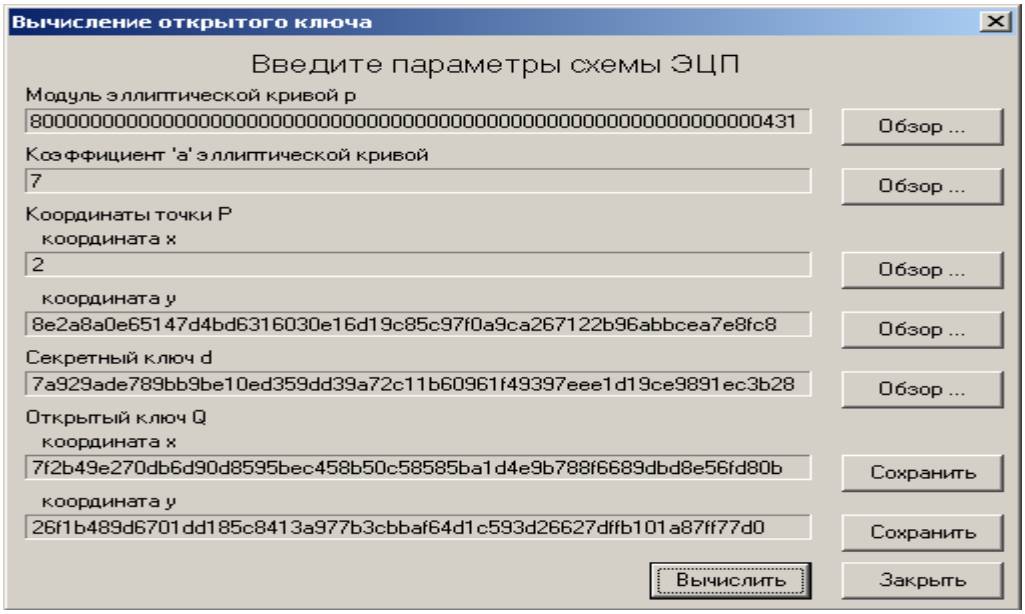


Рис. 6.16. Окно “Вычисление открытого ключа”

Пользователь имеет возможность вычислить открытый ключ для имеющегося у него секретного ключа и параметров схемы ЭЦП.

- Формирование ЭЦП.

При выборе данной опции открывается окно “Формирование ЭЦП” (рисунок 6.17).

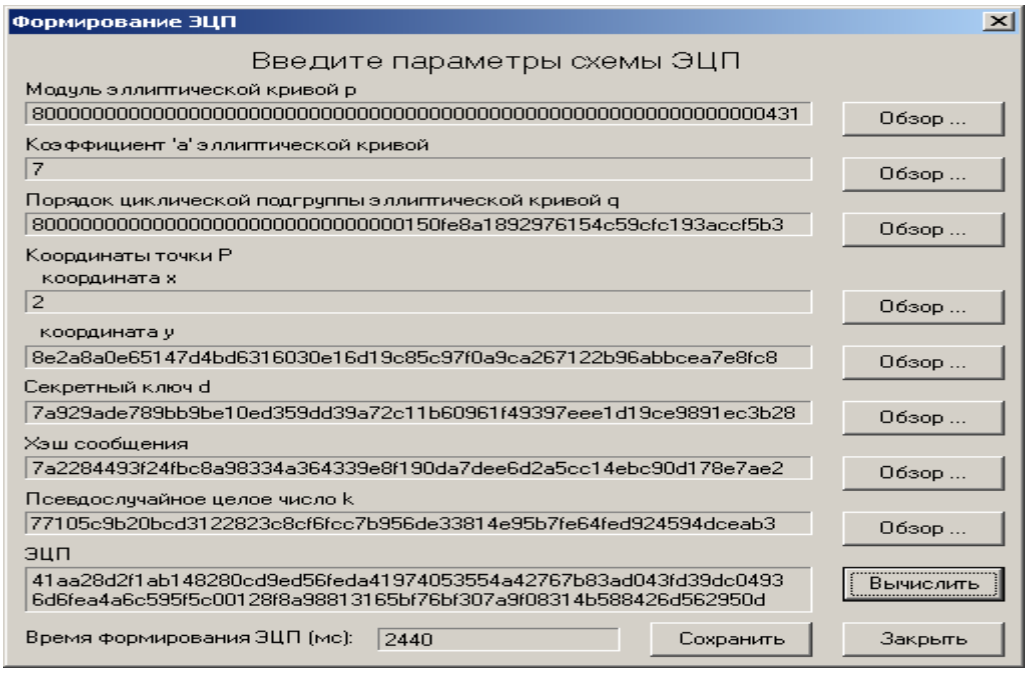


Рис. 6.17. Окно “Формирование ЭЦП”

Пользователь имеет возможность, после ввода всех необходимых для расчетов параметров, вычислить цифровую подпись и сохранить полученный результат в файл.

- Проверка ЭЦП.

При выборе данной опции открывается окно “Проверка ЭЦП” (рисунок 6.18).

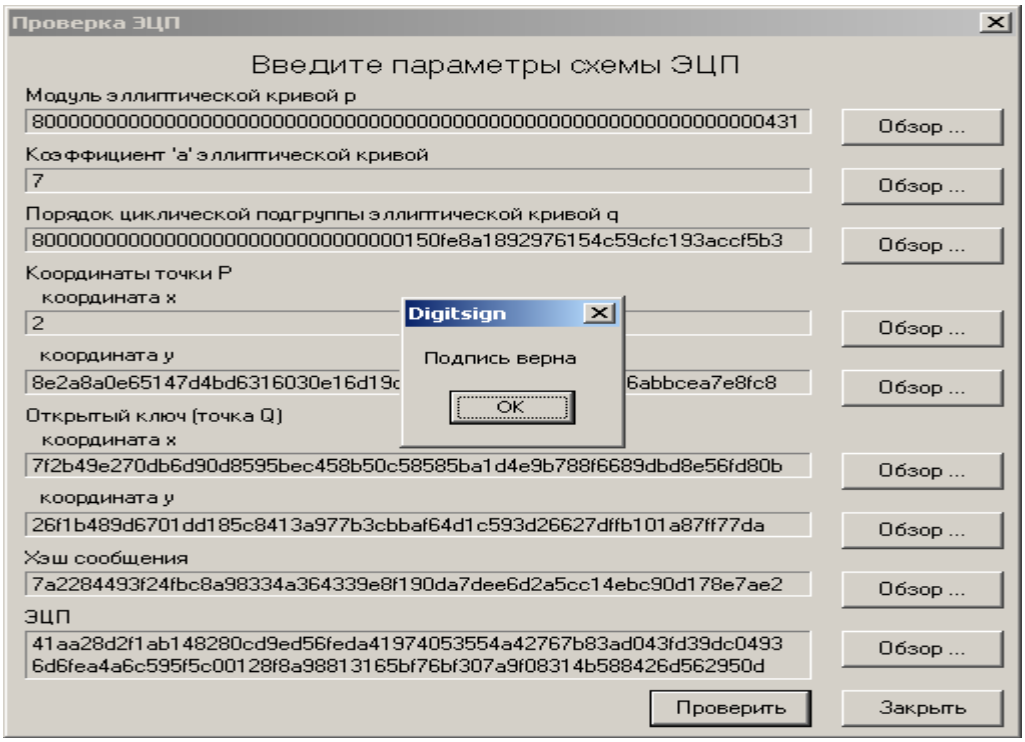


Рис. 6.18. Окно “Проверка ЭЦП”

Пользователь имеет возможность, после ввода всех необходимых для расчетов параметров, проверить подлинность цифровой подписи.

Пункт меню “Справка” содержит следующие опции:

- Задание на лабораторную работу.

Данная опция позволяет открыть файл справки, содержащий задание на лабораторную работу.

- Справка.

Данная опция позволяет открыть файл справки, содержащий информацию о хэш-функции и цифровых подписях.

- О программе.

Пункт меню “Выход” позволяет выйти из программы.

Порядок выполнения работы

Во время выполнения лабораторной работы делайте скриншоты. После вставить в отчет.

1. Ознакомьтесь с теорией по функциям хэширования, эллиптическим кривым и цифровым подписям.

2. Ознакомьтесь с ГОСТ Р 34.11-94 и ГОСТ Р 34.10-2001.

3. Откройте папку DigitSign и запустите программу DigitSign.exe.

4. Изучение функции хэширования по ГОСТ Р 34.11-94.

4.1 Открываем пункт меню «Вычисления» и выбираем «Вычисление хэш-функции».

4.2 Задать начальные данные для вычисления хэш-функции (файлы находятся в папке «Хэширование»):

- путь к файлу, который подвергается сжатию (название text0.txt);
- начальный хэш-вектор (HeshVHex.txt);
- таблицу замен (TabZamen.txt).

4.3 Вычислить хэш-функцию.

4.4 Повторить пункты 4.2 – 4.3 для файлов с названием text1, text2, text3, text4, text5. Сохранить полученные значения хэш-функции в папку «Временное», назвать hash1.txt, hash2.txt, hash3.txt, hash4.txt, hash5.txt соответственно. Определить время хэширования. Полученные результаты занести в первые две строки таблицы (l – размер текстового файла, t_h – время хэширования, для text0 делать данный пункт не обязательно). ВНИМАНИЕ, хэширование может производиться до 3х минут, лучше в этот момент не чего не делать на компьютере, иначе возможно получение недостоверных значений.

l, Мбайт					
t_h, мс					
$t_{ЭЦП}$, мс					
t_{Σ}, мс					

5. Изучение цифровой подписи ГОСТ Р 34.10-2001.

5.1 Вычислить секретный ключ.

5.1.1 Открываем пункт меню «Вычисления» и выбираем «Генератор псевдослучайных чисел».

5.1.2 Ввести параметры датчика (файлы находятся в папке «ГПСЧ»):

- начальное значение X_0 (X0.txt);
- множитель a (a.txt);
- приращение c (c.txt);
- модуль m (m.txt).

5.1.3 Вычислить псевдослучайное число, оно же является секретным ключом.

5.1.4 Сохранить полученное значение в папку «Временное», назвав «KeySekr.txt».

5.2 Вычислить открытый ключ.

5.2.1. Открываем пункт меню «Вычисления» и выбираем “Вычисление открытого ключа”.

5.2.2. Задать параметры, необходимые для вычисления (файлы находятся в папке «Открытый ключ»).

- Модуль эллиптической кривой p (modP.txt);
- Коэффициент эллиптической кривой a (kofA.txt);
- Координаты точки P (координата x – Px.txt, координата y – Py.txt);
- Секретный ключ d (ранее полученный в пункте 5.1.4).

5.2.3. Вычислить открытый ключ.

5.2.4. Сохранить полученное значение в папку «Временное», назвав «Qx.txt» для координаты X и «Qy.txt» для координаты Y.

5.3 Вычислить цифровую подпись.

5.3.1. Выполнить предварительное вычисление случайного числа *k*, необходимого для формирования цифровой подписи. Для этого следует выполнить п. 5.1. Сохранить результат в папку «Временное», назвав «psK.txt».

5.3.2. Открываем пункт меню «Вычислить» и выбираем «Формирование цифровой подписи».

5.3.3. Задать параметры, необходимые для вычисления (файлы находятся в папке «Открытый ключ»):

- Модуль эллиптической кривой p (modP.txt);
- Коэффициент эллиптической кривой a (kofA.txt);
- Порядок циклической подгруппы эллиптической кривой q (q.txt);
- Координаты точки P (координата X – Px.txt, координата Y – Py.txt);
- Секретный ключ d (ранее полученный в пункте 5.1.4);
- Хэш сообщения (ранее полученное в пункте 4.4, с названием hash1.txt);
- Псевдо случайное число k (ранее полученное в пункте 5.4.1).

5.3.4. Сформировать цифровую подпись и занести в таблицу время создания ЭЦП (строка $t_{ЭЦП}$).

5.3.5. Сохранить значение ЭЦП в папку «Временное», назвав «podpis.txt»

5.3.6. Повторить пункты 5.4.3-5.4.4 для хэш-значений hash2.txt; hash3.txt; hash4.txt; hash5.txt. Обратите внимание на то, что пункт 5.4.5 выполнять повторно не нужно.

5.3.7. Рассчитать суммарное время вычисления ЭЦП и занести результаты расчетов в таблицу ($t_{\Sigma}= t_h+ t_{ЭЦП}$).

5.4 Передайте другому студенту, который сидит за другим компьютером, открытый ключ (Qx.txt; Qy.txt), файл (text1.txt) и соответствующую ему цифровую подпись (podpis.txt) при помощи дискеты или сети и получите от него такой же набор. Полученные файлы переименовать:

- «Qx.txt» в «QxPol.txt»;
- «Qy.txt» в «QyPol.txt»;
- «text1.txt» в «text1Pol.txt»;
- «podpis.txt» в «podpisPol.txt»;
- И поместить в папку «Временное».

5.5 Проверить цифровую подпись.

5.5.1 Сделать пункт 4, для файла text1Pol.txt, результат сохранить в папку «Временное», назвав «hashPol.txt»;

5.5.2 Открываем пункт меню «Вычислить», и выбираем «Проверка цифровой подписи».

5.5.3 Задать параметры, необходимые для вычисления (файлы находятся в папке «Открытый ключ»):

- Модуль эллиптической кривой p (modP.txt);
- Коэффициент эллиптической кривой a (kofA.txt);
- Порядок циклической подгруппы эллиптической кривой q (q.txt);
- Координаты точки P (координата X – «Px.txt», Y – «Py.txt»);
- Открытый ключ Q (полученный в пункте 5.4, координата X – «QxPol.txt»; координата Y – «QyPol.txt»);
- Хэш сообщения (поученный в пункте 5.4.1, «hashPol.txt»);
- ЭЦП (полученное в пункте 5.4, «podpisPol.txt»).

5.5.4 Осуществить проверку.

ВНИМАНИЕ! Данная программа неверно вычисляет открытый ключ Q, координату y. Если «Подпись не верна»: открываете файл QyPol.txt через блокнот и добавляете к **последней** цифре единицу (программа все время выдает последнюю цифру равную 0) и снова проверяете, если опять неверна, то еще добавляете единицу до тех пор пока подпись не будет верна или пока не достигнете f и подпись по прежнему не верна, тогда подпись действительно не верна (программа использует 16-ти ричную систему счисления, {0; 1; 2; 3; 4; 5; 6; 7; 8; 9; a; b; c; d; e; f}).

5.6 Подделка сообщения.

5.6.1. Откройте файл «text1Pol.txt» (полученный в пункте 5.4) с помощью блокнота и добавьте, удалите или измените любой символ в тексте, после сохраните изменение в этот же файл.

5.6.2. Повторите пункт 5.5.

6. По полученным результатам построить два графика: на первом графике зависимость времени вычисления хэш-значения от размера файла $t_h = f(l)$, на втором графике зависимость $t_{ЭЦП} = f(l)$.

7. По проделанной работе сделать выводы.

8. Оформить отчет по проделанной работе.

9. Очистить папку «Временное».

Контрольные вопросы

1. Что такое асимметричная криптографическая система? В чем ее преимущества перед симметричной системой? В чем недостатки?
2. Что такое хэш-функция? Какими свойствами она должна обладать и в чем они заключаются?
3. Опишите процесс формирования хэш-функции ГОСТ Р 34.11-94.
4. Что такое “функция сжатия внутренних итераций”? В чем она заключается?
5. Что такое цифровая подпись? В чем ее отличие от рукописной подписи?
6. Опишите процесс формирования цифровой подписи ГОСТ Р 34.10-2001.
7. Опишите процесс проверки цифровой подписи ГОСТ Р 34.10-2001.
8. От каких злоумышленных действий позволяет защититься ЭЦП?

Выполнение работы

10. Изучение функции хэширования по ГОСТ Р 34.11-94.

1. Открываем пункт меню «Вычисления» и выбираем «Вычисление хэш-функции».

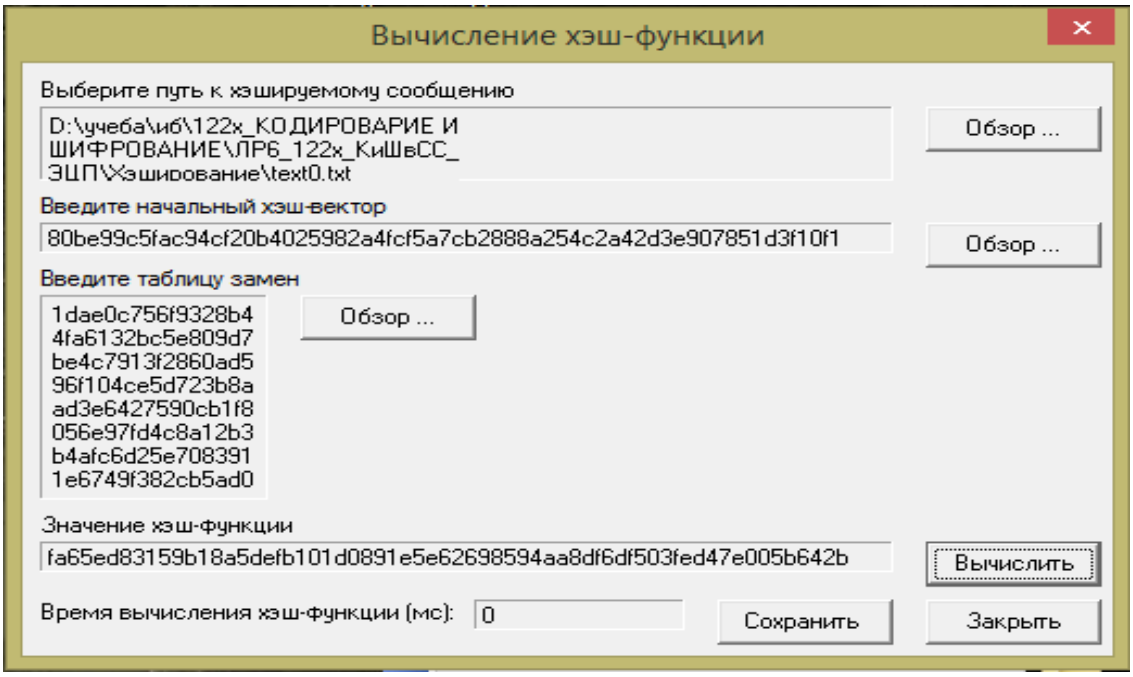
2. Задать начальные данные для вычисления хэш-функции (файлы находятся в папке «Хэширование»):

3. путь к файлу, который подвергается сжатию (название text0.txt);

4. начальный хэш-вектор (HeshVHex.txt);

5. таблицу замен (TabZamen.txt).

6. Вычислить хэш-функцию.



7. Повторить пункты 4.2 – 4.3 для файлов с названием text1, text2, text3, text4, text5. Сохранить полученные значения хэш-функции в папку «Временное», назвать hash1.txt, hash2.txt, hash3.txt, hash4.txt, hash5.txt соответственно. Определить время хэширования. Полученные результаты занести в первые две строки таблицы (l – размер текстового файла, t_h – время хэширования, для text0 делать данный пункт не обязательно). ВНИМАНИЕ, хэширование может производиться до 3х минут, лучше в этот момент не чего не делать на компьютере, иначе возможно получение недостоверных значений.

l, байт	64	64	64	64	64
t_h, мс	15550	31390	46490	63940	78150
$t_{ЭЦП}$, мс	540	530	550	540	550
t_Σ, мс	16090	31920	47040	64480	78700

11. Изучение цифровой подписи ГОСТ Р 34.10-2001.

5.2 Вычислить секретный ключ.

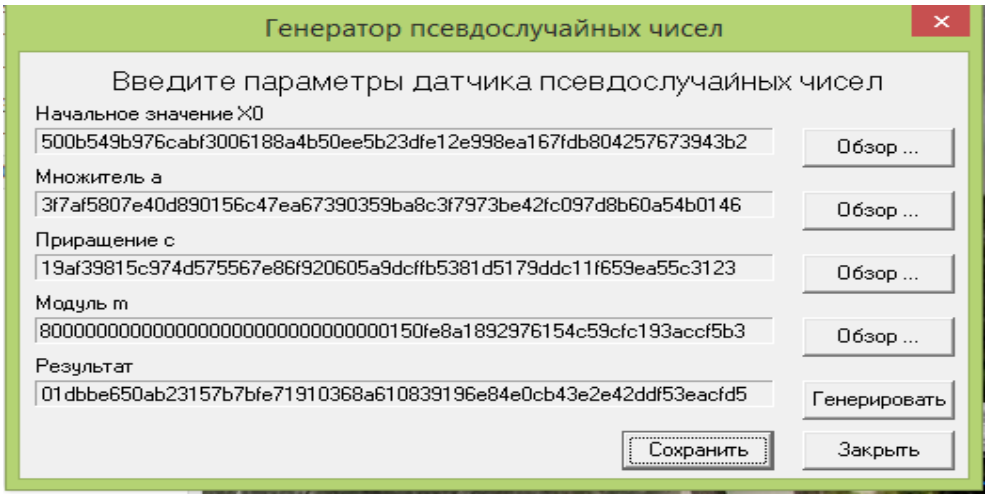
5.1.5 Открываем пункт меню «Вычисления» и выбираем «Генератор псевдослучайных чисел».

5.1.6 Ввести параметры датчика (файлы находятся в папке «ГПСЧ»):

- начальное значение X_0 (X0.txt);
- множитель a (a.txt);
- приращение c (c.txt);
- модуль m (m.txt).

5.1.7 Вычислить псевдослучайное число, оно же является секретным ключом.

5.1.8 Сохранить полученное значение в папку «Временное», назвав «KeySekr.txt».



5.3 Вычислить открытый ключ.

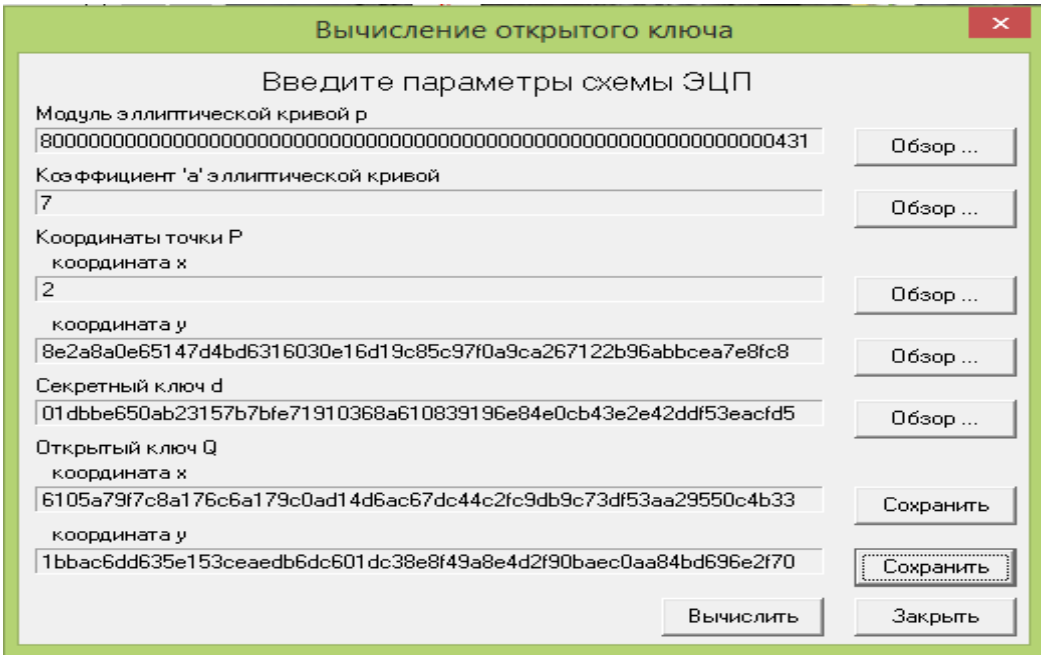
5.2.5. Открываем пункт меню «Вычисления» и выбираем “Вычисление открытого ключа”.

5.2.6. Задать параметры, необходимые для вычисления (файлы находятся в папке «Открытый ключ»).

- Модуль эллиптической кривой p (modP.txt);
- Коэффициент эллиптической кривой a (kofA.txt);
- Координаты точки P (координата x – Px.txt, координата y – Py.txt);
- Секретный ключ d (ранее полученный в пункте 5.1.4).

5.2.7. Вычислить открытый ключ.

5.2.8. Сохранить полученное значение в папку «Временное», назвав «Qx.txt» для координаты X и «Qy.txt» для координаты Y.



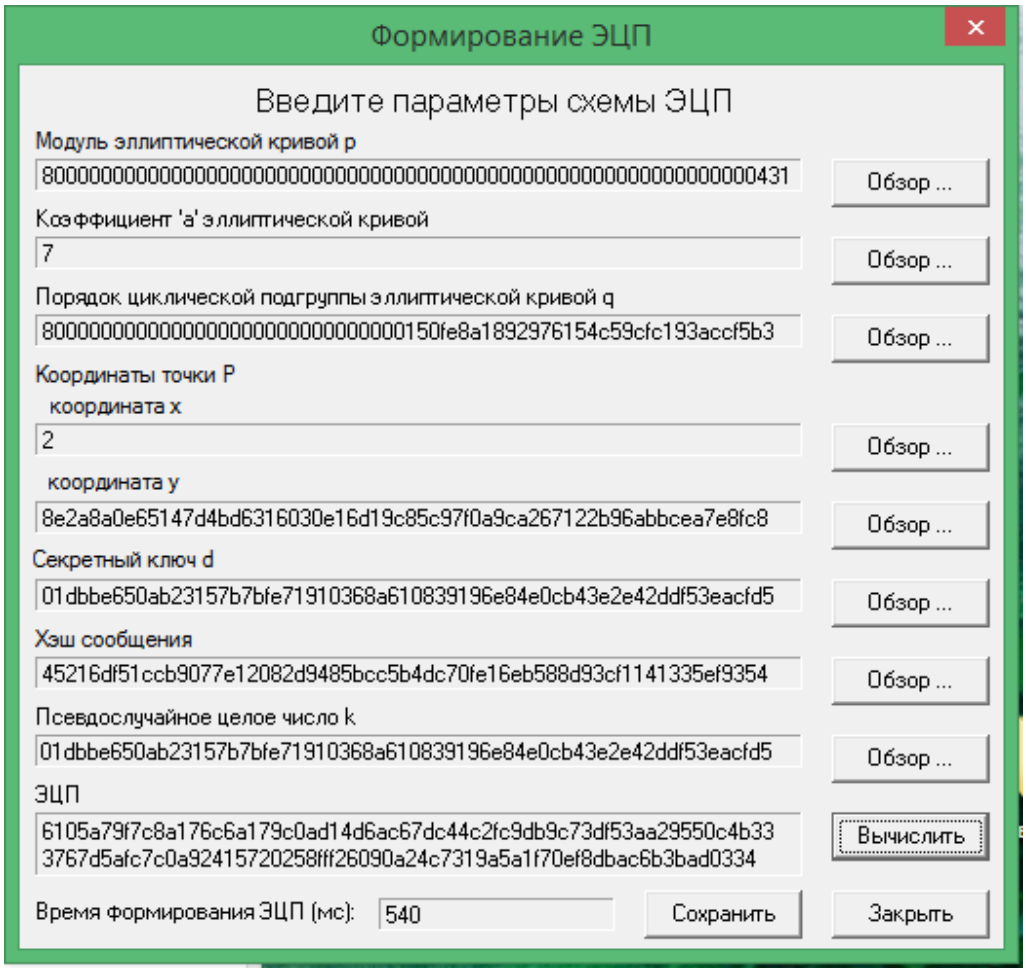
6.19. Вычислить цифровую подпись.

5.3.1. Выполнить предварительное вычисление случайного числа *k*, необходимого для формирования цифровой подписи. Для этого следует выполнить п. 5.1. Сохранить результат в папку «Временное», назвав «psK.txt».

5.3.2. Открываем пункт меню «Вычислить» и выбираем «Формирование цифровой подписи».

5.3.3. Задать параметры, необходимые для вычисления (файлы находятся в папке «Открытый ключ»):

- Модуль эллиптической кривой p (modP.txt);
- Коэффициент эллиптической кривой a (kofA.txt);
- Порядок циклической подгруппы эллиптической кривой q (q.txt);
- Координаты точки P (координата X – Px.txt, координата Y – Py.txt);
- Секретный ключ d (ранее полученный в пункте 5.1.4);
- Хэш сообщения (ранее полученное в пункте 4.4, с названием hash1.txt);
- Псевдо случайное число k (ранее полученное в пункте 5.4.1).



5.3.4. Сформировать цифровую подпись и занести в таблицу время создания ЭЦП (строка $t_{ЭЦП}$).

5.3.5. Сохранить значение ЭЦП в папку «Временное», назвав «podpis.txt»

5.3.6. Повторить пункты 5.4.3-5.4.4 для хэш-значений hash2.txt; hash3.txt; hash4.txt; hash5.txt. Обратите внимание на то, что пункт 5.4.5 выполнять повторно не нужно.

5.3.7. Рассчитать суммарное время вычисления ЭЦП и занести результаты расчетов в таблицу ($t_{\Sigma} = t_h + t_{ЭЦП}$).

5.6 Передайте другому студенту, который сидит за другим компьютером, открытый ключ (Qx.txt; Qy.txt), файл (text1.txt) и соответствующую ему цифровую подпись (podpis.txt) при помощи дискеты или сети и получите от него такой же набор. Полученные файлы переименовать:

- «Qx.txt» в «QxPol.txt»;
- «Qy.txt» в «QyPol.txt»;
- «text1.txt» в «text1Pol.txt»;
- «podpis.txt» в «podpisPol.txt»;
- И поместить в папку «Временное».

5.7 Проверить цифровую подпись.

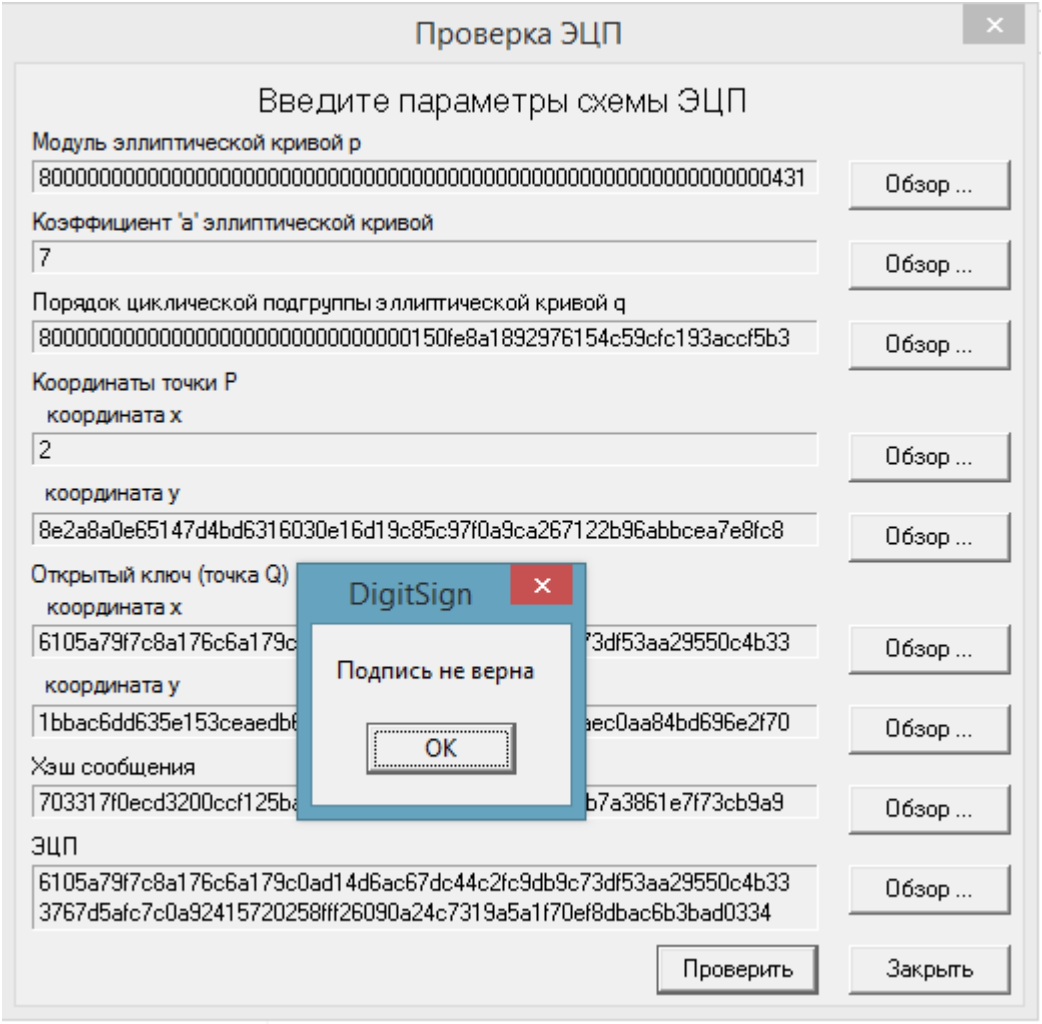
5.5.5 Сделать пункт 4, для файла text1Pol.txt, результат сохранить в папку «Временное», назвав «hashPol.txt»;

5.5.6 Открываем пункт меню «Вычислить», и выбираем «Проверка цифровой подписи».

5.5.7 Задать параметры, необходимые для вычисления (файлы находятся в папке «Открытый ключ»):

- Модуль эллиптической кривой p (modP.txt);
- Коэффициент эллиптической кривой a (kofA.txt);
- Порядок циклической подгруппы эллиптической кривой q (q.txt);
- Координаты точки P (координата X – «Px.txt», Y – «Py.txt»);
- Открытый ключ Q (полученный в пункте 5.4, координата X – «QxPol.txt»; координата Y – «QyPol.txt»);
- Хэш сообщения (поученный в пункте 5.4.1, «hashPol.txt»);
- ЭЦП (полученное в пункте 5.4, «podpisPol.txt»).

5.5.8 Осуществить проверку.



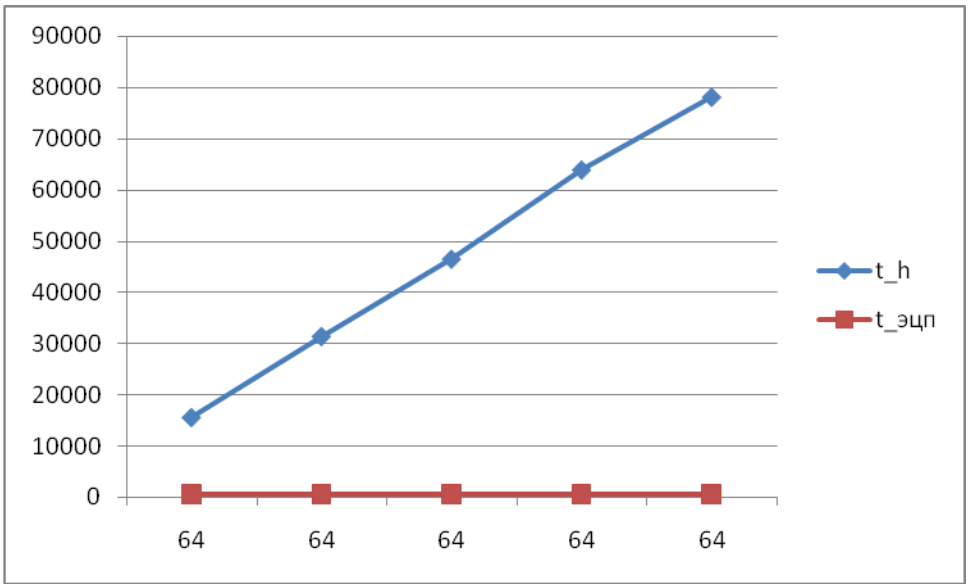
ВНИМАНИЕ! Данная программа неверно вычисляет открытый ключ Q, координату y. Если «Подпись не верна»: открываете файл QyPol.txt через блокнот и добавляете к **последней** цифре единицу (программа все время выдает последнюю цифру равную 0) и снова проверяете, если опять неверна, то еще добавляете единицу до тех пор пока подпись не будет верна или пока не достигнете f и подпись по прежнему не верна, тогда подпись действительно не верна (программа использует 16-ти ричную систему счисления, {0; 1; 2; 3; 4; 5; 6; 7; 8; 9; a; b; c; d; e; f}).

5.7 Подделка сообщения.

5.6.3. Откройте файл «text1Pol.txt» (полученный в пункте 5.4) с помощью блокнота и добавьте, удалите или измените любой символ в тексте, после сохраните изменение в этот же файл.

5.6.4. Повторите пункт 5.5.

12. По полученным результатам построить два графика: на первом графике зависимость времени вычисления хэш-значения от размера файла $t_h = f(l)$, на втором графике зависимость $t_{ЭЦП} = f(l)$.



Вывод: В результате выполнения данной лабораторной работы мы изучили процессы вычисления функции хэширования по ГОСТ Р 34.11-94, формирования и проверки ЭЦП на эллиптических кривых по ГОСТ Р 34.10-2001.

Лабораторная работа 7. Исследование алгоритмов PGP кодирования и шифрования с открытым ключом

PGP (англ. Pretty Good Privacy) — компьютерная программа, также библиотека функций, позволяющая выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, в том числе прозрачное шифрование данных на запоминающих устройствах, например, на жёстком диске.

Первоначально разработана Филиппом Циммерманном в 1991 году.

Общие сведения

PGP имеет множество реализаций, совместимых между собой и рядом других программ (GnuPG, FileCrypt и др.) благодаря стандарту OpenPGP (RFC 4880), но имеющих разный набор функциональных возможностей. Существуют реализации PGP для всех наиболее распространённых операционных систем. Кроме свободно распространяемых реализаций есть еще и коммерческие.

Совместимость

Так как PGP развивается, некоторые системы позволяют создавать зашифрованные сообщения с использованием новых возможностей, которые отсутствуют в старых системах. Отправитель и получатель должны знать возможности друг друга или, по крайней мере, согласовать настройки PGP.

Защищённость

В 1996 году криптограф Брюс Шнайер охарактеризовал раннюю версию PGP как «ближайшую к криптосистемам военного уровня». На данный момент не известно ни одного способа взлома данных, зашифрованных PGP, при помощи полного перебора или уязвимости криптоалгоритма. Ранние версии PGP обладали теоретическими уязвимостями, поэтому рекомендуется пользоваться современными версиями.

Криптографическая стойкость PGP основана на предположении, что используемые алгоритмы устойчивы к криптоанализу на современном оборудовании. Например, в PGP первых версий для шифрования ключей сессии использовался алгоритм RSA, основанный на односторонней функции (факторизация). В PGP версии 2 дополнительно можно использовать алгоритм IDEA. В последующем были добавлены дополнительные алгоритмы шифрования. Ни у одного используемого алгоритма нет известных уязвимостей.

В 2010 году группе учёных из Швейцарии, Японии, Франции, Нидерландов, Германии и США удалось декодировать данные, зашифрованные по алгоритму RSA при помощи ключа длиной 768 бит. Нахождение простых сомножителей осуществлялось общим методом

решета числового поля. На первый шаг (выбор пары полиномов степени 6 и 1) было потрачено около полугода вычислений на 80 процессорах, что составило около 3 % времени, потраченного на главный этап алгоритма (просеивание), который выполнялся на сотнях компьютеров в течение почти двух лет. Если интерполировать это время на работу одного процессора AMD Opteron 2.2ГГц с 2Гб памяти, то получилось бы порядка 1500 лет. Обработка данных после просеивания для следующего ресурсоёмкого шага (линейной алгебры) потребовала несколько недель на малом количестве процессоров. Заключительный шаг после нахождения нетривиальных решений ОСЛУ занял не более 12 часов.

Решение ОСЛУ проводилось с помощью метода Видемана на нескольких раздельных кластерах и длилось чуть менее 4 месяцев. При этом размер разреженной матрицы составил 192 796 550x192 795 550 при наличии 27 795 115 920 ненулевых элементов. Для хранения матрицы на жёстком диске понадобилось около 105 гигабайт. В то же время понадобилось около 5 терабайт сжатых данных для построения данной матрицы.

В итоге группе удалось вычислить 232-цифровой ключ, открывающий доступ к зашифрованным данным.

Исследователи уверены, что с использованием их метода факторизации взломать 1024-битный RSA-ключ будет возможно в течение следующего десятилетия.

По словам исследователей, после их работы в качестве надежной системы шифрования можно рассматривать только RSA-ключи длиной 1024 бита и более. Причём от шифрования ключом длиной в 1024 бит стоит отказаться в ближайшие три-четыре года.

Зная разложение модуля на произведение двух простых чисел, противник может легко найти секретную экспоненту и тем самым взломать RSA. Однако на сегодняшний день самый быстрый алгоритм факторизации — решето обобщённого числового поля (General Number Field Sieve), скорость которого для k-битного целого числа составляет $\exp((c+o(1))k^{\frac{1}{3}}\log^{\frac{2}{3}}k)$ для некоторого $c<2$, не позволяет разложить большое целое за приемлемое время.

Механизм работы PGP

Шифрование PGP осуществляется последовательно хешированием, сжатием данных, шифрованием с симметричным ключом, и, наконец, шифрованием с открытым ключом, причём каждый этап может осуществляться одним из нескольких поддерживаемых алгоритмов. Симметричное шифрование производится с использованием одного из семи симметричных алгоритмов (AES, CAST5, 3DES, IDEA, Twofish, Blowfish, Camellia) на сеансовом ключе. Сеансовый ключ генерируется с использованием криптографически стойкого генератора псевдослучайных чисел. Сеансовый ключ зашифровывается открытым

ключом получателя с использованием алгоритмов RSAили Elgamal (в зависимости от типа ключа получателя). Каждый открытый ключ соответствует имени пользователя или адресу электронной почты. Первая версия системы называлась Сеть Доверия и противопоставлялась системе X.509, использовавшей иерархический подход, основанной на удостоверяющих центрах, добавленный в PGP позже. Современные версии PGP включают оба способа.

Ключи

Пользователь PGP создаёт ключевую пару: открытый и закрытый ключ. При генерации ключей задаются их владелец (имя и адрес электронной почты), тип ключа, длина ключа и срок его действия. Открытый ключ используется для шифрования и проверки цифровой подписи. Закрытый ключ - для декодирования и создания цифровой подписи.

PGP поддерживает три типа ключей RSA v4, RSA legacy (v3) и Diffie-Hellman/DSS (Elgamal в терминологии GnuPG).

Для ключей RSA legacy длина ключа может составлять от 1024 до 2048 бит, а для Diffie-Hellman/DSS и RSA — от 1024 до 4096. Ключи RSA legacy содержат одну ключевую пару, а ключи Diffie-Hellman/DSS и RSA могут содержать один главный ключ и дополнительные ключи для шифрования. При этом ключ электронной подписи в ключах Diffie-Hellman/DSS всегда имеет размер 1024. Срок действия для каждого из типов ключей может быть определён как неограниченный или до конкретной даты. Для защиты ключевого контейнера используется секретная фраза.

Цифровая подпись

PGP поддерживает аутентификацию и проверку целостности посредством цифровой подписи. По умолчанию она используется совместно с шифрованием, но также может быть применена и к открытому тексту. Отправитель использует PGP для создания подписи алгоритмом RSA или DSA. При этом сначала создаётся хеш открытого текста (также известный как дайджест), затем — цифровая подпись хеша при помощи закрытого ключа отправителя. Для формирования хеша могут использоваться алгоритмы MD5, SHA-1, RIPEMD-160, SHA-256, SHA-384, SHA-512. В новых версиях PGP поддержка MD5 осуществляется для сохранения совместимости с ранними версиями. Для подписи используются алгоритмы RSA или DSA (в зависимости от типа ключа).

Сжатие данных

В целях уменьшения объёма сообщений и файлов и, возможно, для затруднения криптоанализа PGP производит сжатие данных перед шифрованием. Сжатие производится по одному из алгоритмов ZIP, ZLIB, BZIP2. Для сжатых, коротких и слабосжимаемых файлов сжатие не выполняется.

Сеть доверия

Как при шифровании сообщений, так и при проверке цифровой подписи, необходимо, чтобы принятый получателем открытый ключ действительно принадлежал отправителю. При простом скачивании открытого ключа он может быть подменён. С первых версий PGP поддерживает сертификаты открытых ключей, с помощью которых подмены (или случайные ошибки передачи) легко распознаются. Однако недостаточно просто создать сертификат, защищённый от модификации, так как при этом гарантируется лишь целостность сертификата после его создания. Пользователи также должны каким-нибудь способом проверить, что открытый ключ в сертификате действительно принадлежит отправителю. С первых версий продукты PGP включают в себя внутреннюю схему проверки сертификатов, названную сеть доверия. Заданная пара «имя пользователя — открытый ключ» может быть подписана третьим лицом, удостоверяющим соответствие ключа и владельца. В таких подписях может быть несколько вложенных уровней доверия. Хотя многие программы читают и пишут эту информацию, очень немногие учитывают этот уровень сертификата, принимая решение о принятии или отклонении сертификата.

Протокол сети доверия был впервые описан Циммерманном в 1992 году в руководстве PGP версии 2.0: «С течением времени вы будете накапливать ключи других людей, которых вы можете назвать доверенными рекомендателями. Кто-нибудь ещё может выбрать своих доверительных рекомендателей. И все будут постепенно накапливать и распространять со своими ключами набор заверенных подписей других людей, ожидая, что любой получатель доверяет по крайней мере одной или двум подписям. Это позволяет создать децентрализованную устойчивую к сбоям сеть всех открытых ключей.»

Механизм сети доверия обладает преимуществами над централизованной инфраструктурой управления открытыми ключами, например, используемой в S/MIME, но не получил повсеместного применения. Пользователи хотели проверять корректность сертификатов вручную или не проверять вовсе.

Сертификаты

В последних спецификациях OpenPGP доверенные подписи могут использоваться для поддержки создания центров сертификации. Доверенность сертификата означает, что ключ действительно принадлежит указанному владельцу и может использоваться для подписи сертификатов одним уровнем ниже. Сертификат уровня 0 означает обычную подпись. Уровень 1 означает, что при помощи подписанного ключа можно создавать сертификаты уровня 0. При помощи сертификата уровня 2 можно создавать сертификаты уровня 1. Уровень 2 практически идентичен степени доверия, с которой полагаются пользователи на списки доверенных сертификатов, встроенные в браузеры.

Все версии PGP включают в себя способ отмены сертификата. Это необходимо, если требуется сохранять безопасность связи при потере или компрометации закрытого ключа. Отмена сертификата похожа на списки отзыва сертификатов в централизованной инфраструктуре открытых ключей. Современные версии PGP также поддерживают сроки истечения сертификатов.

Проблема корректного определения принадлежности открытого ключа владельцу характерна для всех криптографических систем с асимметричным шифрованием. У неё не существует достаточно хороших решений. Оригинальная схема PGP позволяет решить пользователю, использовать ли схему проверки сертификатов, в то время как большинство других инфраструктур открытых ключей требуют проверки каждого сертификата.

История

В 1991 году Филипп Циммерман создал первую версию PGP. Первая версия включала в себя симметричный алгоритм шифрования BassOmatic, созданный самим Циммерманом. Циммерман участвовал в движении против ядерной энергии и создал PGP для защищённого использования BBS и хранения файлов и сообщений. Для некоммерческого использования не требовалось лицензии, со всеми копиями распространялся весь исходный код. PGP распространилась в Usenet, а затем и в Интернете.

Уголовное расследование

Вскоре после выпуска PGP стала использоваться за пределами США, и в 1993 году правительство США начало расследование против Циммермана по подозрению в нарушении экспортного законодательства, которое регулирует распространение криптографических систем с длиной ключа более 40 бит. В PGP использовались ключи длиной 128 бит и более.

Циммерман остроумно обошёл ограничения законодательства США. Он опубликовал исходный код в книге, изданной MIT Press. Код можно было сосканировать, распознать и скомпилировать. Экспорт книг не может быть запрещён, так как защищён первой поправкой к Конституции США.

OpenPGP

PGP Inc. была обеспокоена по поводу патентов. В компании был создан внутренний стандарт Unencumbered PGP («необременённый PGP»), не использующий алгоритмы, имеющие проблемы с лицензиями. Так как PGP широко использовалась во всём мире, многие хотели создавать собственное ПО, совместимое с PGP 5. В 1997 году PGP Inc. предложила IETF стандарт, названный OpenPGP. В IETF был созданы стандарты RFC 2440 (1998 год) и RFC 4880 (2007 год).

В 1999 году силами Фонда свободного программного обеспечения была создана свободная реализация OpenPGP под названием GNU Privacy Guard (GnuPG).

Поглощение Network Associates

В декабре 1997 года PGP Inc. была поглощена Network Associates Inc (ныне McAfee). NAI продолжила экспорт посредством печати исходных текстов. В составе NAI команда PGP разработала средства для шифрования дисков, брандмауэр, средства для обнаружения вторжений и IPsec VPN. После легализации экспорта криптографического ПО в 2000 году NAI прекратила публикацию исходных текстов, несмотря на возражения команды PGP.

В 2001 году Циммерман покинул NAI, NAI объявила о продаже PGP и остановке разработки PGP. В 2002 году NAI прекратила поддержку всех продуктов PGP PGP E-Business Server (исходной консольной версии PGP).

Современное состояние

В 2002 году несколько бывших разработчиков PGP основали PGP Corporation и выкупили PGP (кроме консольной версии). В 2003 году PGP Corporation разработала новый серверный продукт, PGP Universal.

В 2010-м году Symantec Corp. выкупил PGP за 300 млн долларов.

Криптографические приложения PGP Corporation

PGP изначально разрабатывалась для шифрования электронной почты на стороне клиента, но с 2002 года включает также шифрование жёстких дисков переносных компьютеров, файлов и директорий, сессий программ мгновенного обмена сообщениям, пакетной передачи файлов, защиту файлов и директорий в сетевых хранилищах, а в современных версиях — ещё и шифрование HTTP-запросов и ответов на стороне сервера (mod openpgp) и клиента (Enigform).

Клиентские программы объединены в семейство PGP Desktop (включает в себя PGP Desktop EMail, PGP Whole Disk Encryption и PGP NetShare).

PGP Universal Server позволяет из командной строки централизованно администрировать клиенты на основе PGP Desktop.

В 2010 году права на приложение были приобретены компаний Symantec за 300 млн. долларов.

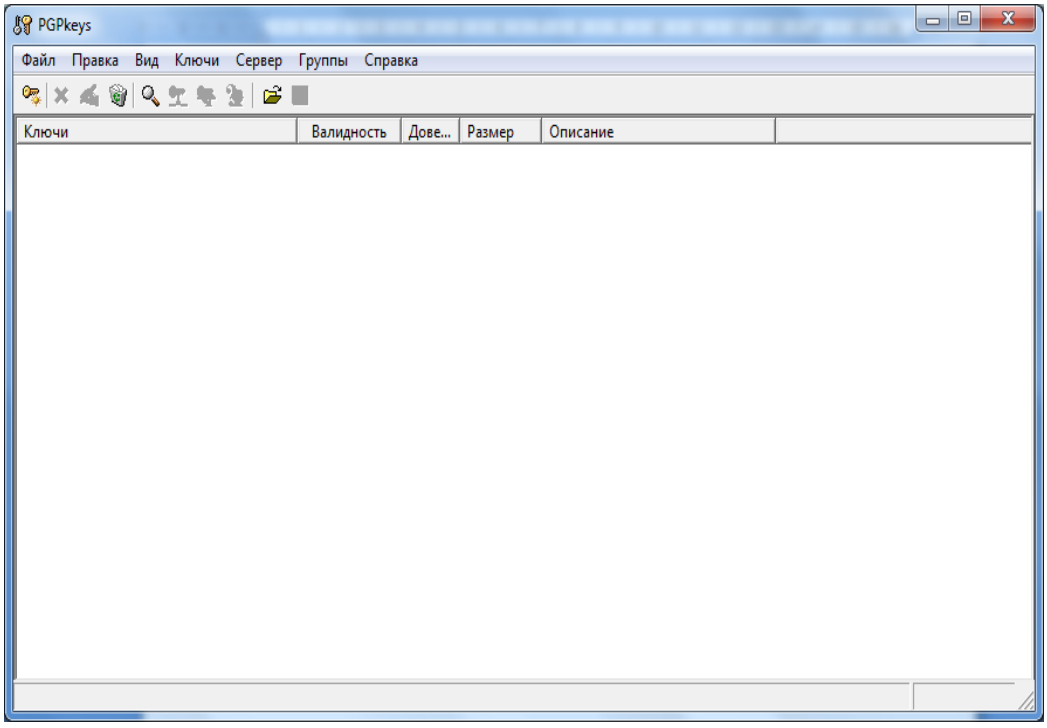
Правовые аспекты использования в России

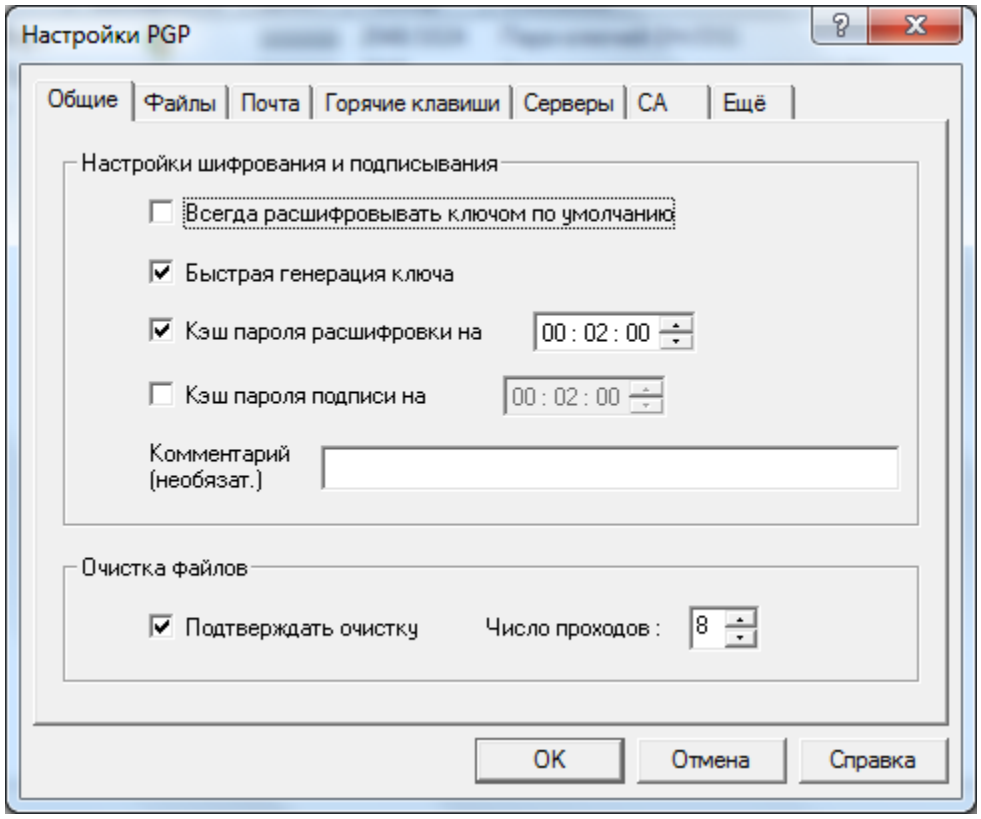
На сегодняшний день прямых законодательных запретов на использование PGP в России нет. Законодательно ограничивается использование криптографии только в государственных и муниципальных учреждениях. ФСБ предписывает всем государственным структурам использовать только сертифицированные средства криптографии. Физические лица и компании сами устанавливают какая информация является для них коммерческой

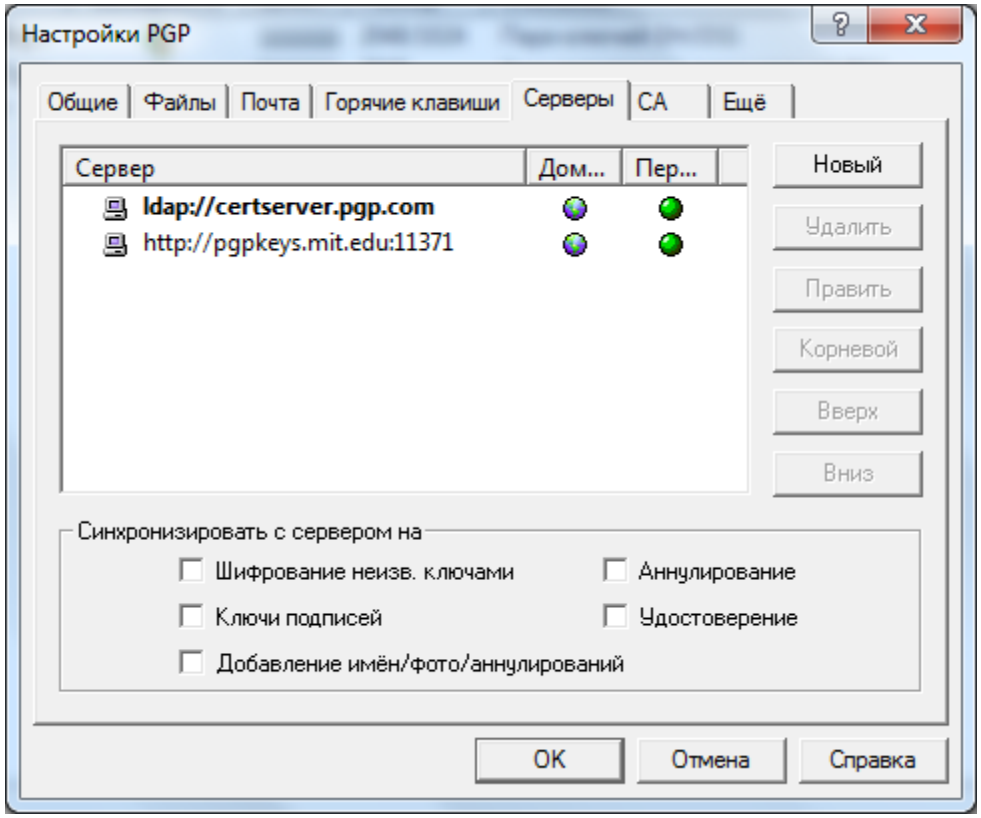
тайной, методы хранения и передачи такой информации Закон «Об информации, информационных технологиях и защите информации» также указывает, что способ защиты информации, представляющей тайну, для негосударственных структур определяется оператором. Информационный ресурс Helpdesk24 в статье «Правомерность использования криптографических средств защиты информации» приводит выдержки из федеральных законов, поясняющие данный вопрос. Также авторы проекта «openPGP в России» утверждают, что не существует законов, запрещающих использование PGP . Указ от 3 апреля 1995 г. N 334 «О мерах по соблюдению законности в области разработки, производства, реализации и эксплуатации шифровальных средств» отменен. Электронная подпись, генерируемая с помощью PGP и её несертифицированных аналогов, имеет юридическую силу в Российской Федерации, т.к. согласно пункту 3 статьи 5 63-ФЗ "Об электронной подписи" попадает под определение усиленной неквалифицированной электронной подписи. Согласно пункту 2 статьи 6 этого ФЗ для признания такой ЭП необходимо соглашение между участниками электронного взаимодействия.

2.ХОД РАБОТЫ

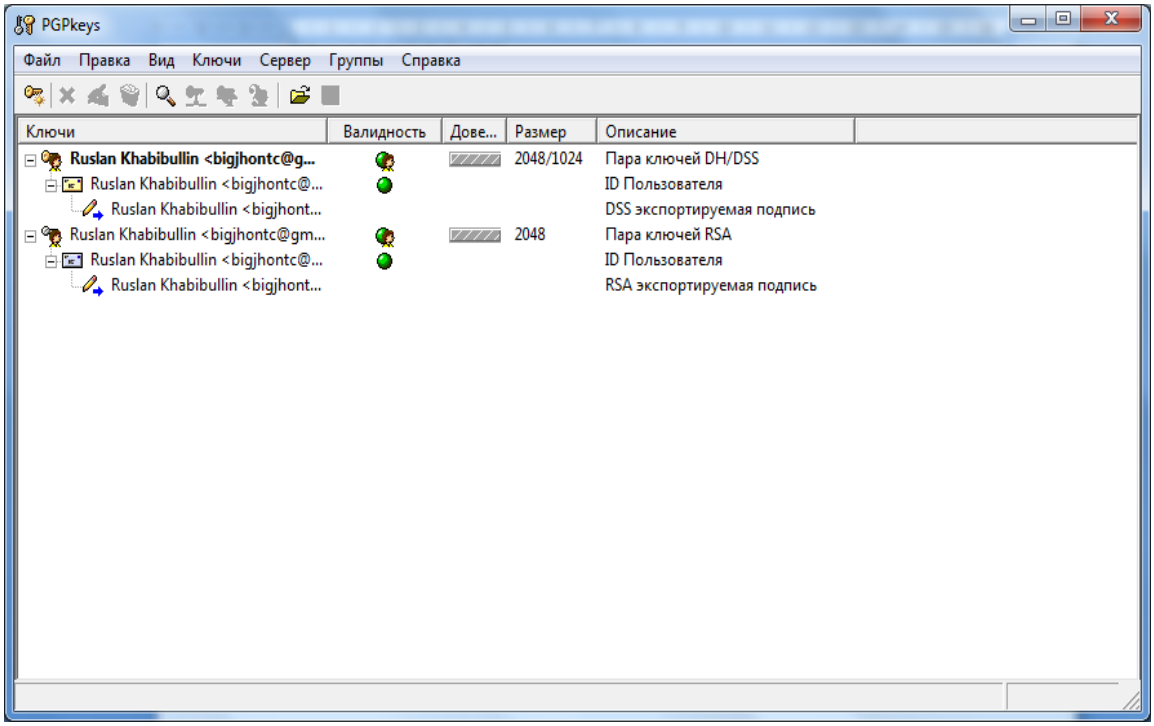
1. Изучите вкладки окна “Настройки PGP”.



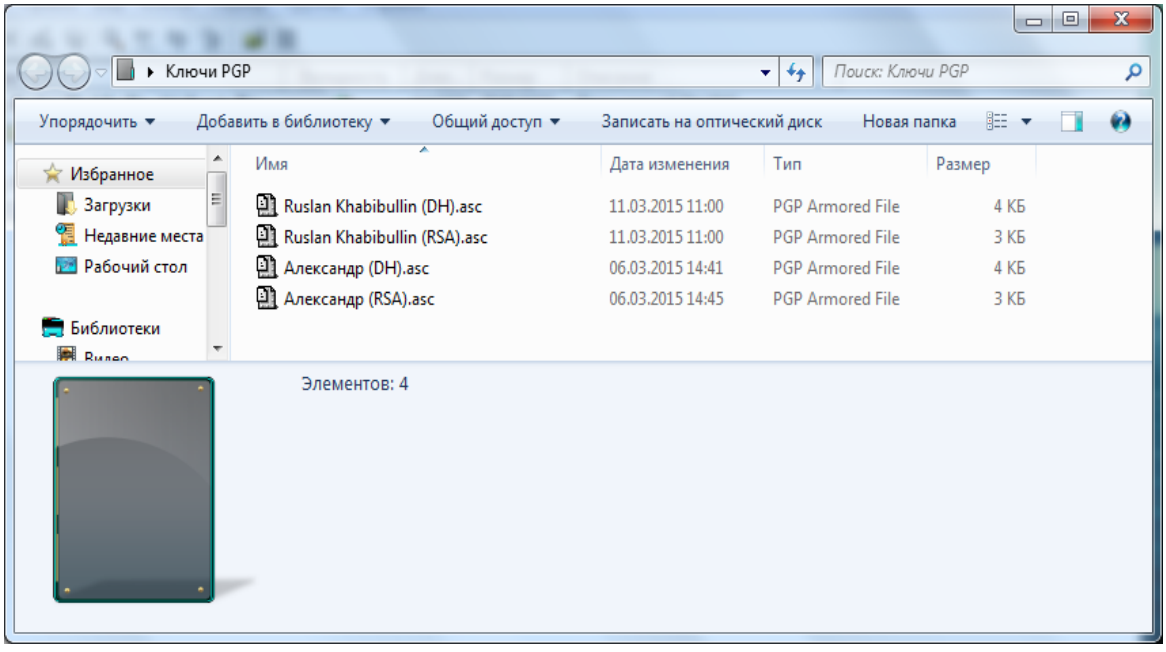

Настройки PGP
Общие
Файлы
Почта
Горячие клавиши
Серверы
CA
Ещё
Настройки шифрования и подписывания
Всегда расшифровывать ключом по умолчанию
Быстрая генерация ключа
Кэш пароля расшифровки на
00 : 02 : 00
Кэш пароля подписи на
00 : 02 : 00
Комментарий (необязат.)
Очистка файлов
Подтверждать очистку
Число проходов :
8
OK
Отмена
Справка


Настройки PGP
Общие
Файлы
Почта
Горячие клавиши
Серверы
CA
Ещё
Сервер
Дом...
Пер...
ldap://certserver.pgp.com
http://pgpkeys.mit.edu:11371
Новый
Удалить
Править
Корневой
Вверх
Вниз
Синхронизировать с сервером на
Шифрование неизв. ключами
Аннулирование
Ключи подписей
Удостоверение
Добавление имён/фото/аннулирований
OK
Отмена
Справка

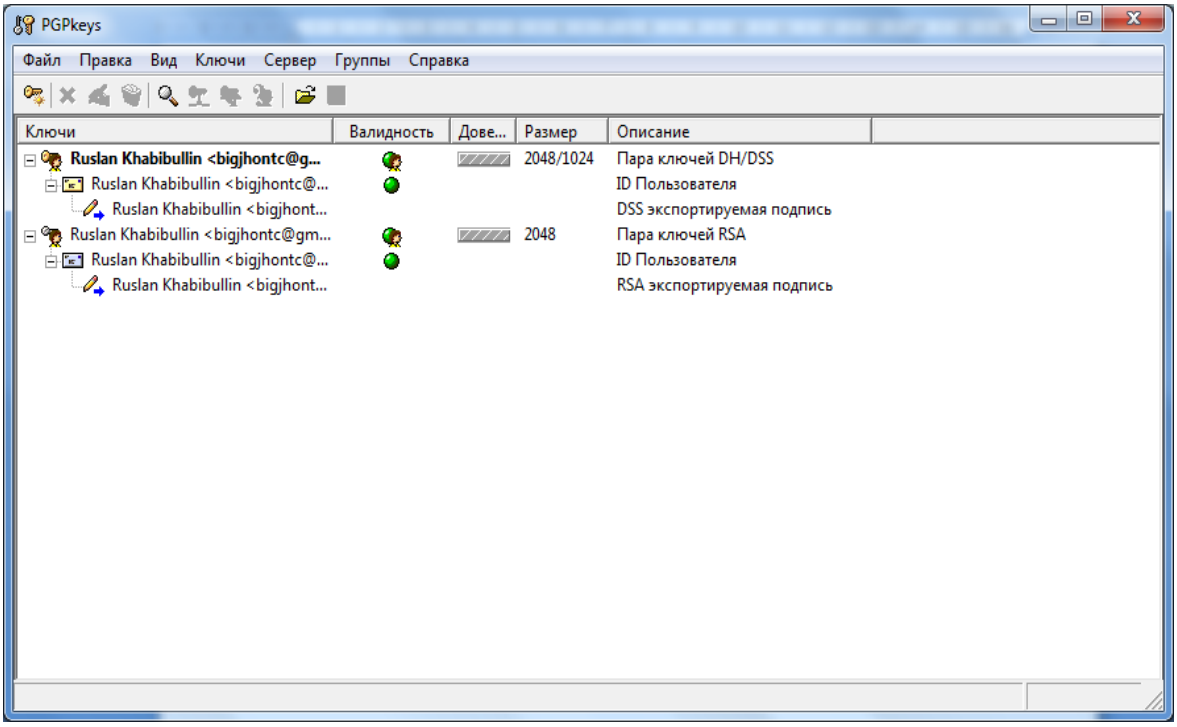
2. Создайте с помощью менеджера PGPkeys ключи шифрования двух типов: RSA и DH/DSS.



3. Сохраните полученные ключи (открытые и секретные) в отдельный файл.

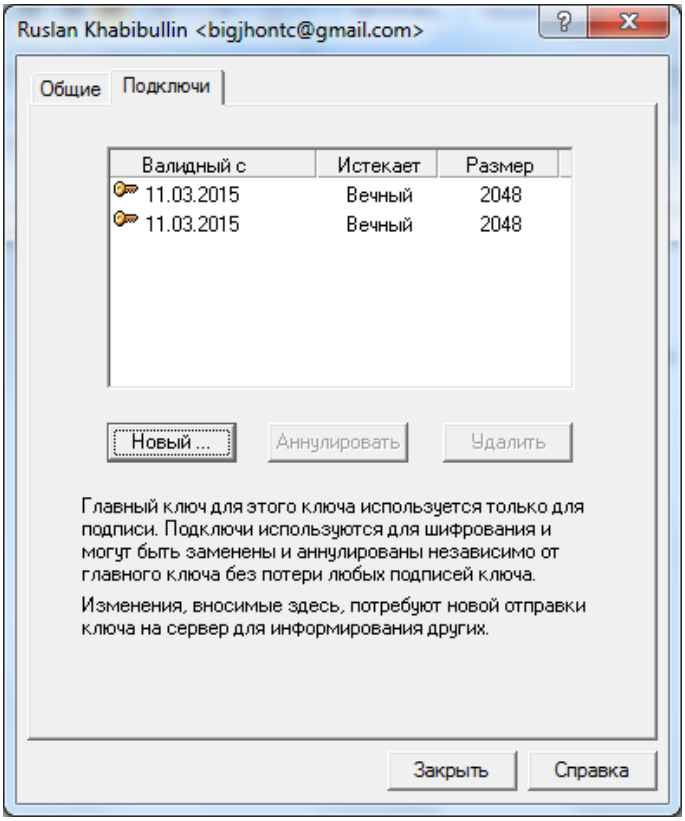


4. Назначьте ключ DH/DSS используемым по умолчанию.

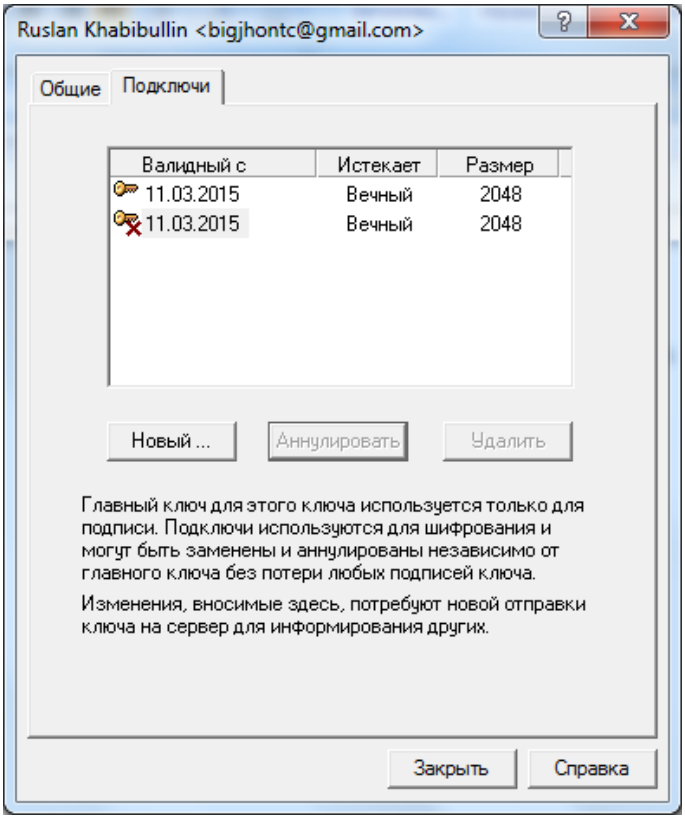


5. Изучите свойства ключевой пары DH/DSS.

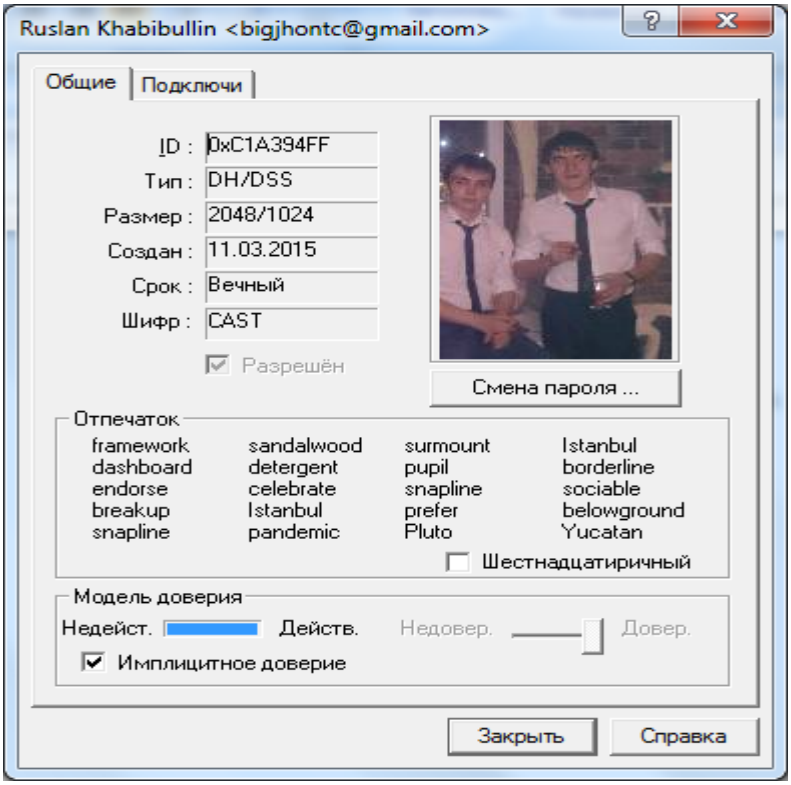
Создание подключа



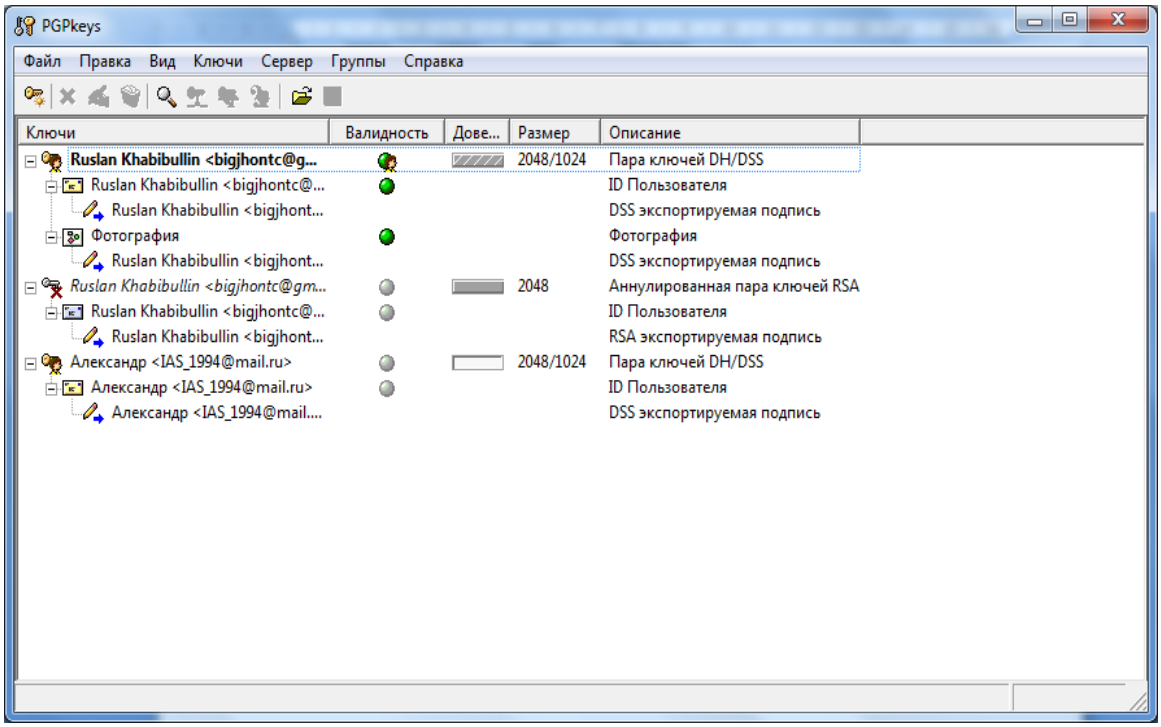
Аннулирование подключа



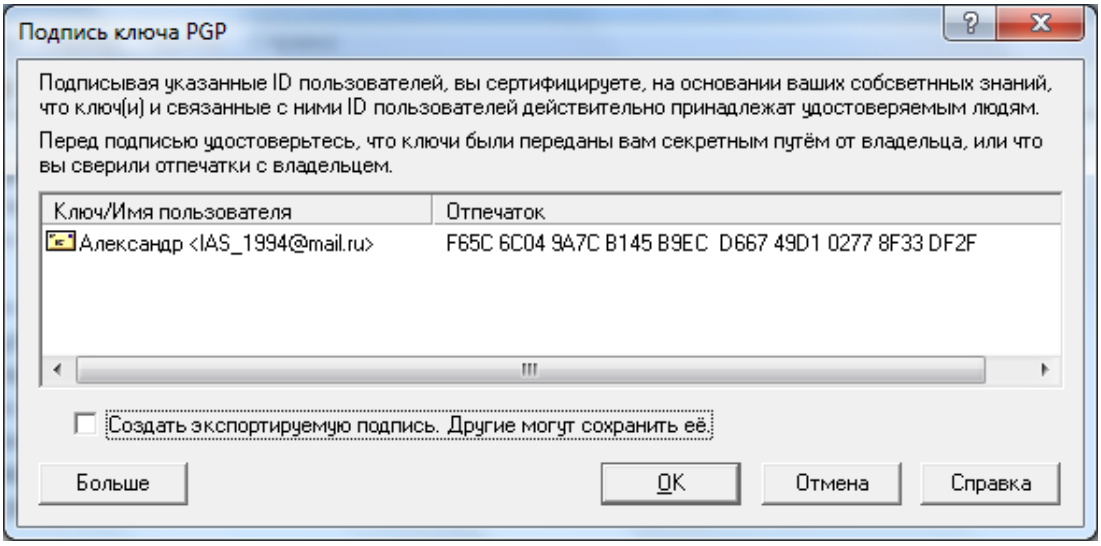
Добавление фотографии



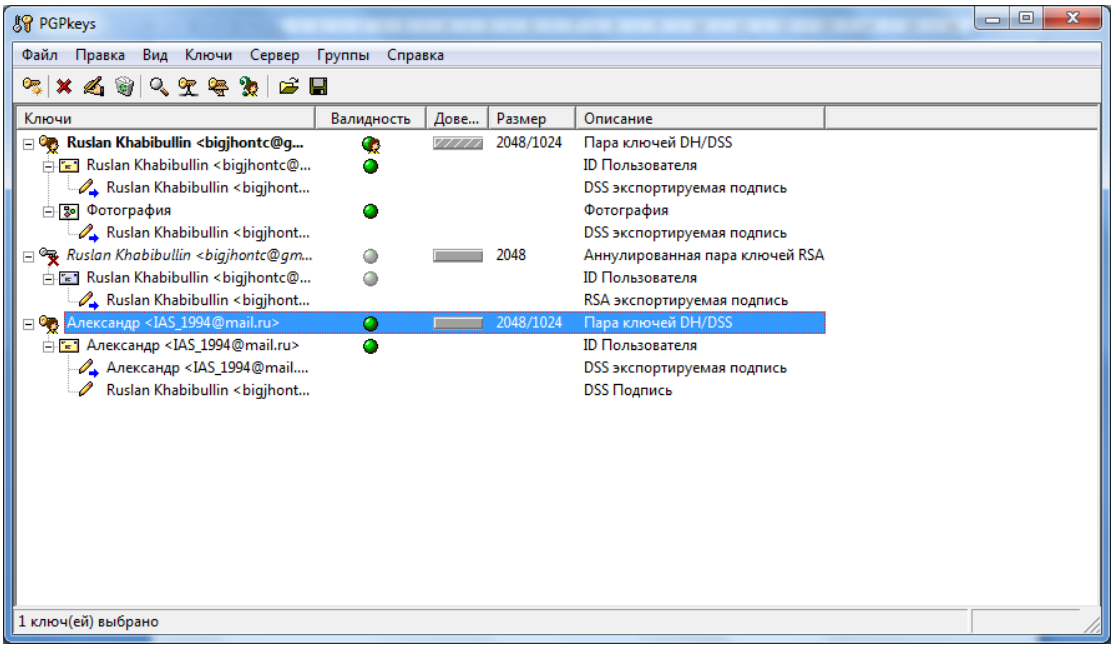
6. Обменяйтесь с другим студентом открытыми ключами DH/DSS.



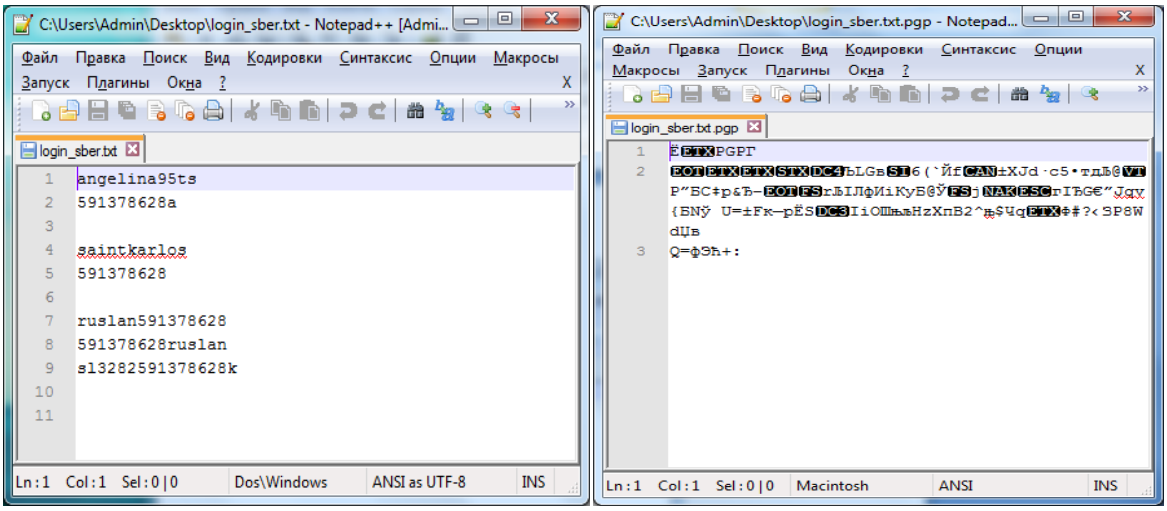
7. Установите подлинность полученного ключа с помощью его отпечатка.

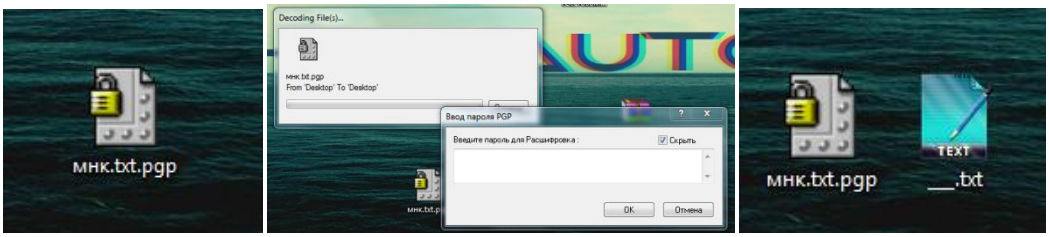


8. Установите степень доверия к владельцу полученного ключа на максимальный уровень.

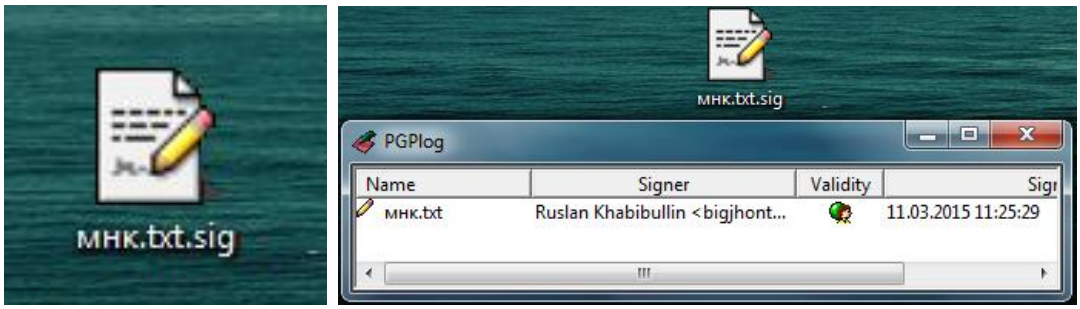


9. Зашифруйте произвольное сообщение/файл и обменяйтесь полученным результатом с другим студентом. Расшифруйте полученное сообщение.

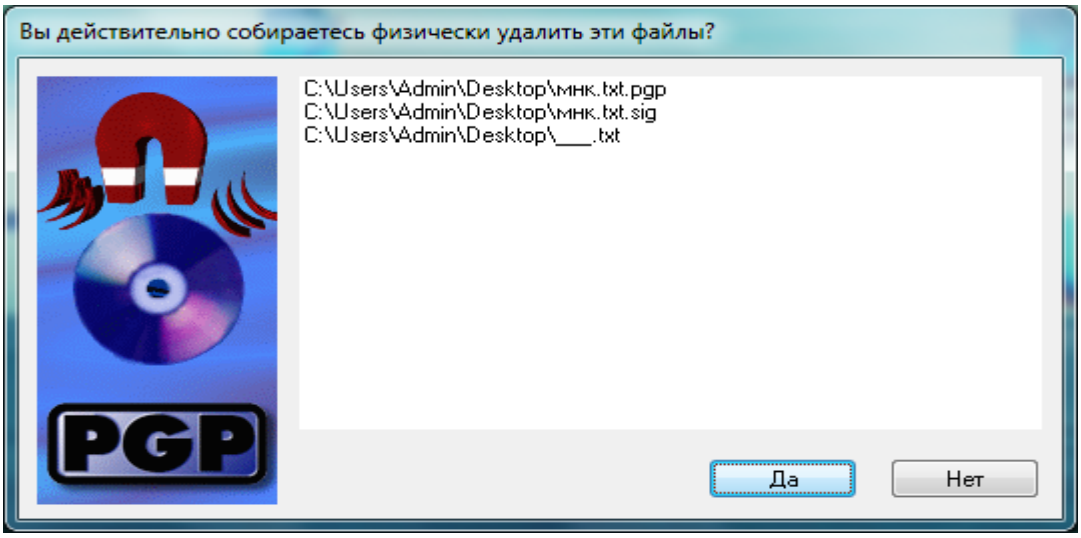




10. Подпишите произвольное сообщение/файл и обменяйтесь полученным результатом с другим студентом. Проверьте достоверность источников полученного сообщения.



11. Уничтожьте все ненужные файлы, используя утилиту PGP Wipe.



Контрольные вопросы.

1. Криптографическая система с открытым ключом (или асимметричное шифрование, асимметричный шифр) — система шифрования и/или электронной подписи (ЭП), при которой открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для проверки ЭП и для шифрования сообщения. Для генерации ЭП и для расшифровки сообщения используется закрытый ключ. Криптографические системы с открытым ключом в настоящее время широко применяются в различных сетевых протоколах, в частности, в протоколах TLS и его предшественнике SSL (лежащих в основе HTTPS), в SSH. Также используется в PGP, S/MIME.

2. Электронная подпись (ЭП), Электронная цифровая подпись (ЭЦП) — реквизит электронного документа, полученный в результате криптографического преобразования информации с использованием закрытого ключа подписи и позволяющий установить отсутствие искажения информации в электронном документе с момента формирования подписи и проверить принадлежность подписи владельцу сертификата ключа подписи.

3. Аннулирование – после аннулирования открытого ключа PGP синхронизирует его с сервером, дабы в дальнейшем ваши корреспонденты не могли его применять. **Дезактивация –** временное отключение неиспользуемого ключа или ключевой пары.

4. Отпечаток ключа — серия знаков, которой сопровождается каждый ключ. Сам по себе он не секретный, но уникальный. Если отпечаток того ключа, который вы получили (например) по электронной почте, и отпечаток, которым поделился ваш друг в Skype, совпадут, значит, у вас в руках правильный, настоящий ключ.

5. Имплицитное доверие – полное доверие зарезервировано для ключевых пар, расположенных на локальном ключе. Если одна часть ключевой пары находится на вашей связке, PGP предполагает, что вы владелец пары ключей и что Вы без проблем сможете доверять себе.

Лабораторна работа. 8 Исследование защищенных сетевых протоколов SSLи TLS

Программная реализация

Для создания защищенного соединения необходимо получить сертификат или создать его самому. Создать SSL сертификат самому можно средствами свободно распространяемого пакета IIS6 Resource Kit Tools. Может потребоваться установка служб. Для этого в Панели управления необходимо выбрать раздел «Программы и компоненты» и в открывшемся окне в меню слева нажать на ссылку «Включение или отключение компонентов Windows». Далее в открывшемся окне нужно включить компонент «Службы IIS».

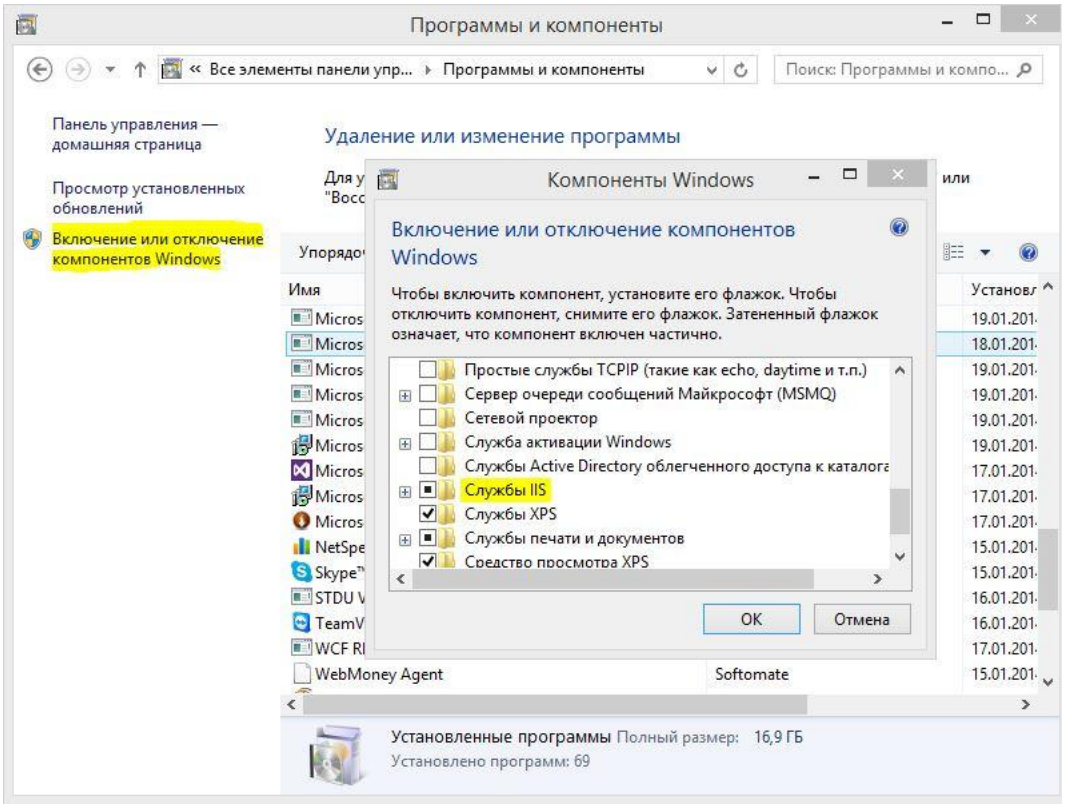


Рис. 8.1. Активация службы ISS

Выбираем службу ISS

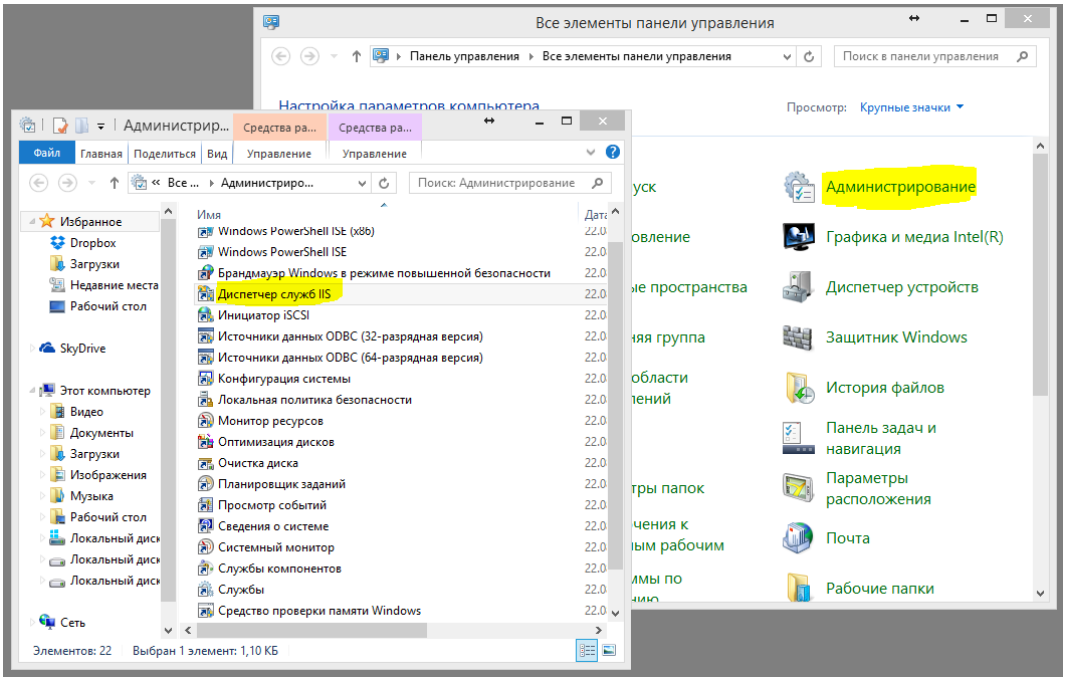


Рис. 8.2. Выбор службы ISS

В открывшемся окне жмем на «Сертификаты сервера»

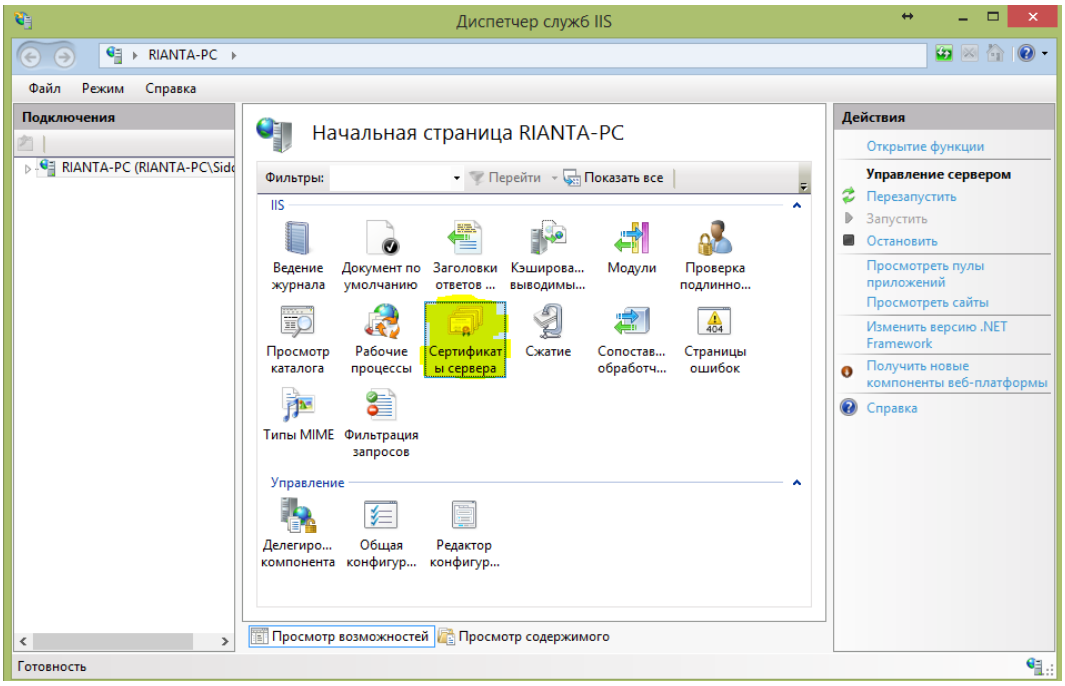


Рис. 8.3. Создание сертификата

Создаем самозаверенный сертификат

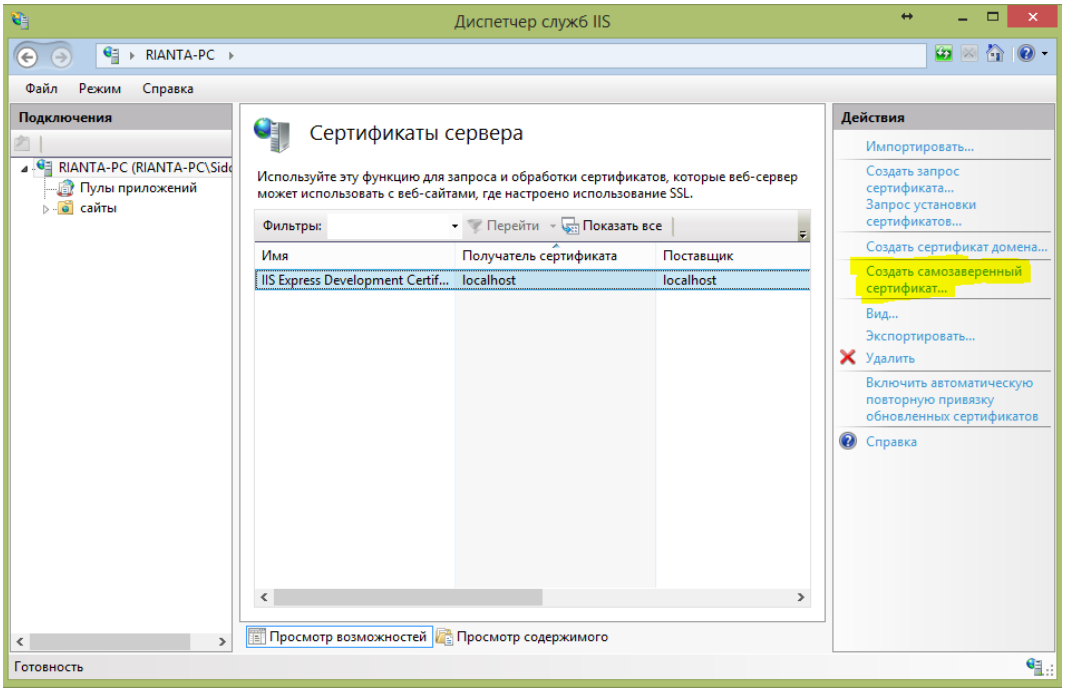


Рис. 8.4. Создание самозаверенного сертификата

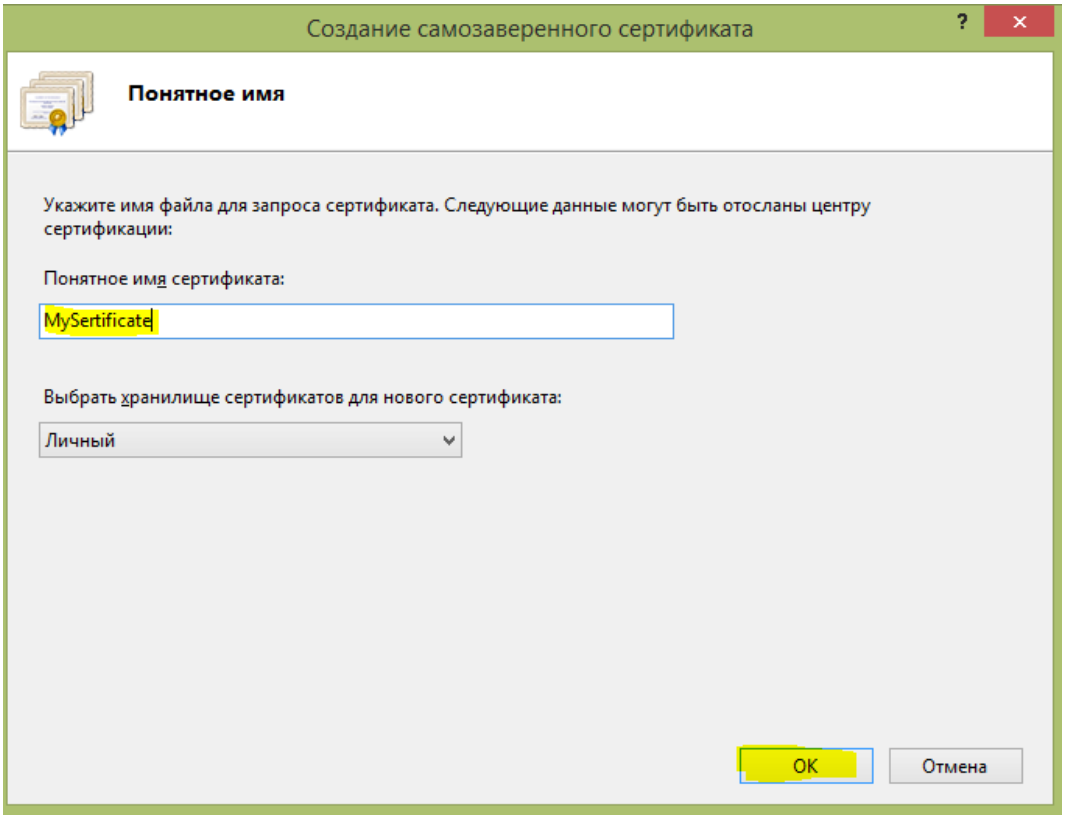


Рис. 8.5. Ввод имени сертификата

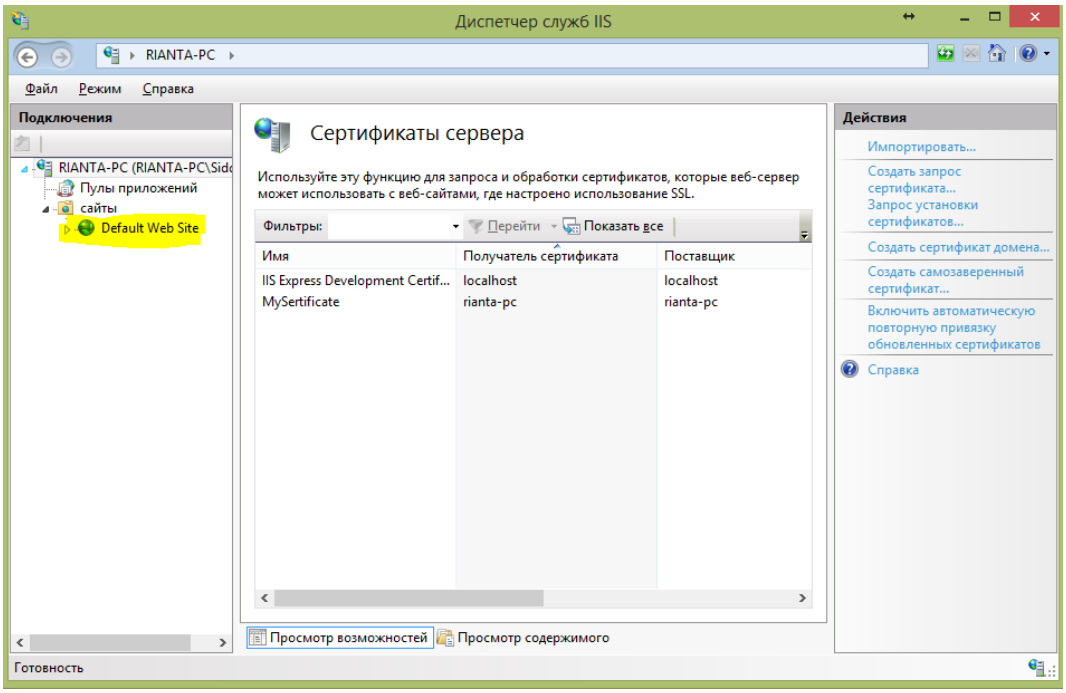


Рис. 8.6. Отображение сертификата

Привязываем сертификат к нужному IP

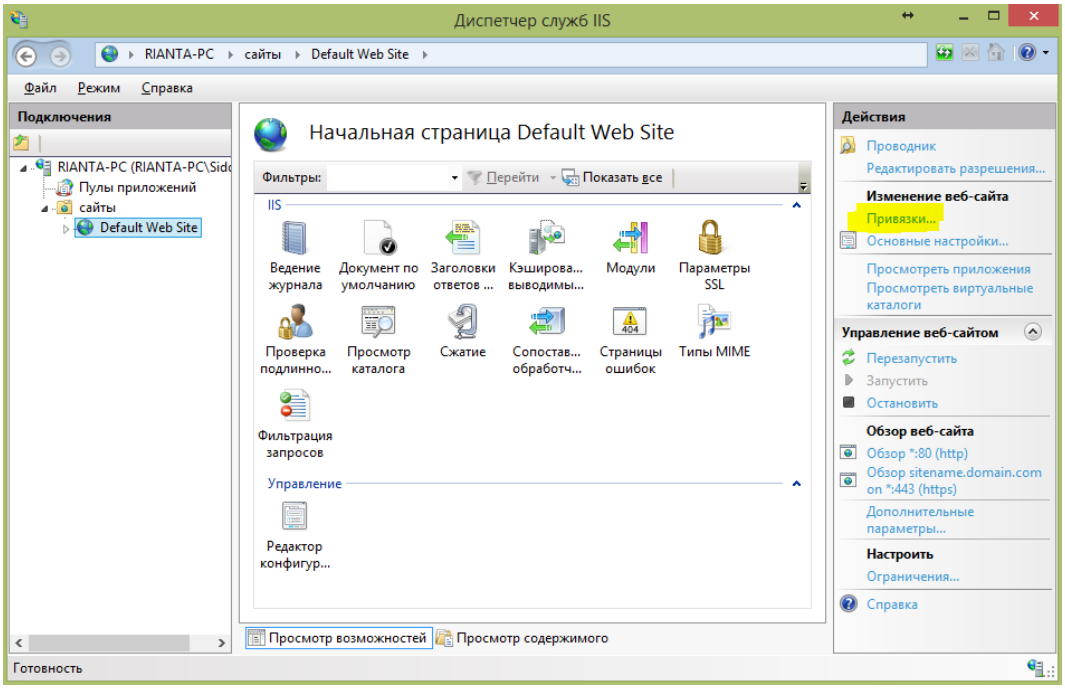


Рис. 8.7. Привязка сертификата к сайту (компьютеру)

Затем с помощью программы THEGREENBOW устанавливаем защищенное SSL – соединение.

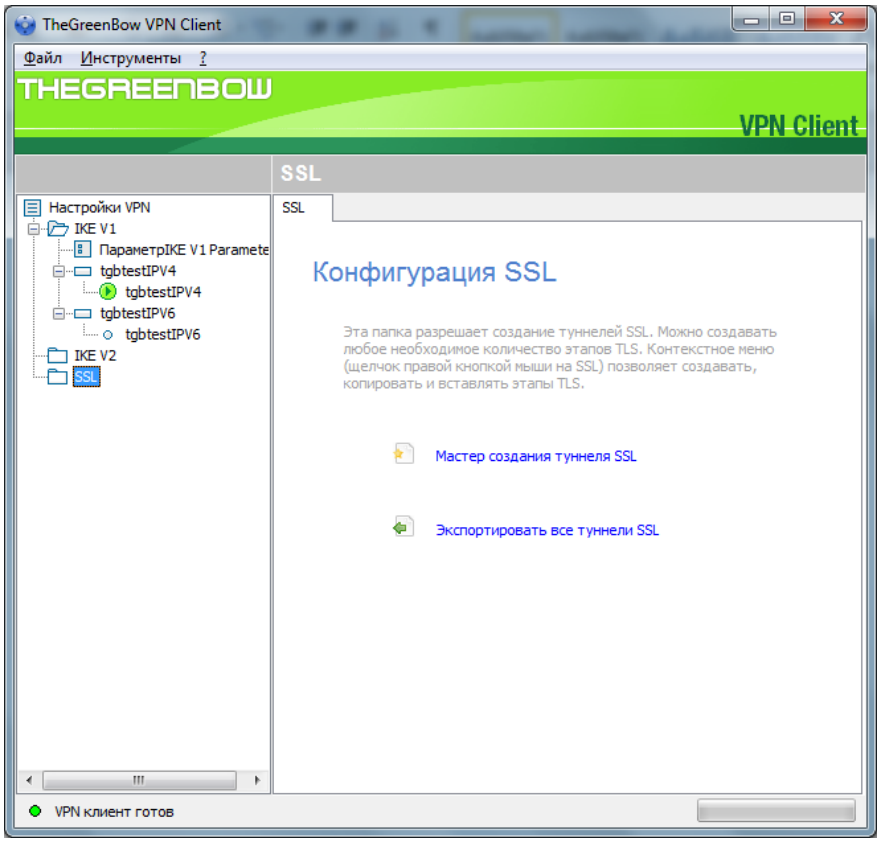


Рис.8.8. Окно раздела SSL

Указываем путь к сертификату

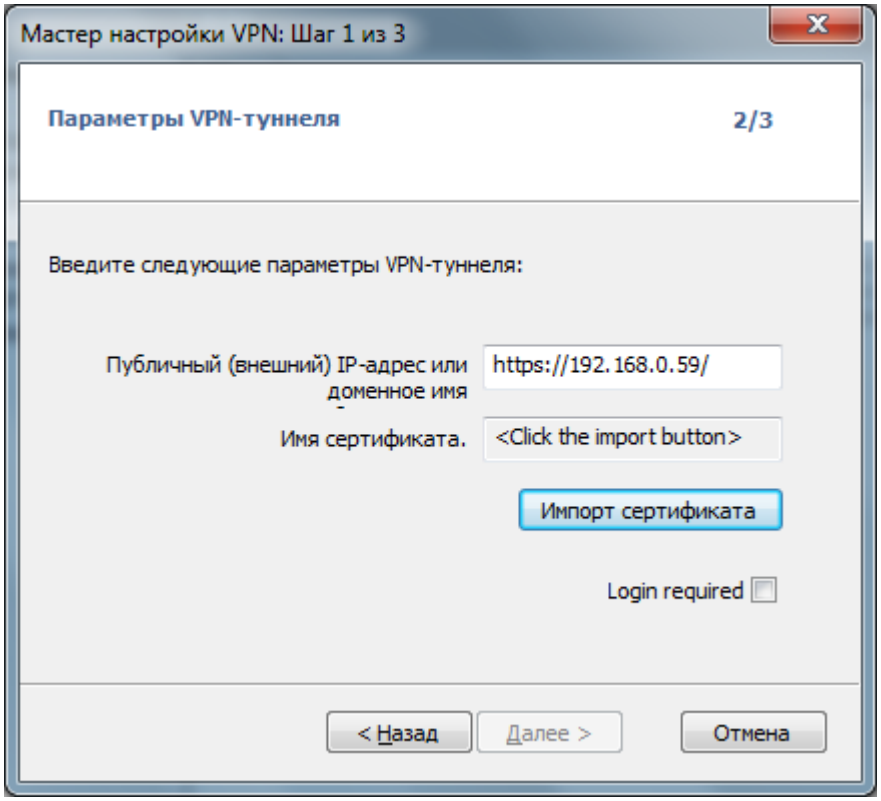


Рис. 8.9. Окно мастера настройки VPN-туннеля

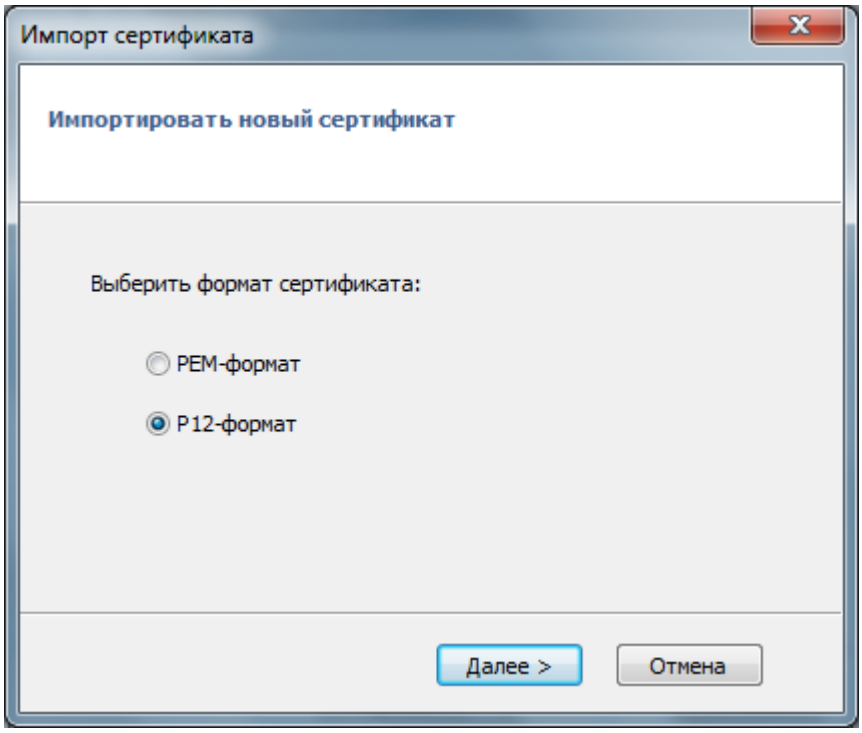


Рис. 8.10. Импорт сертификата

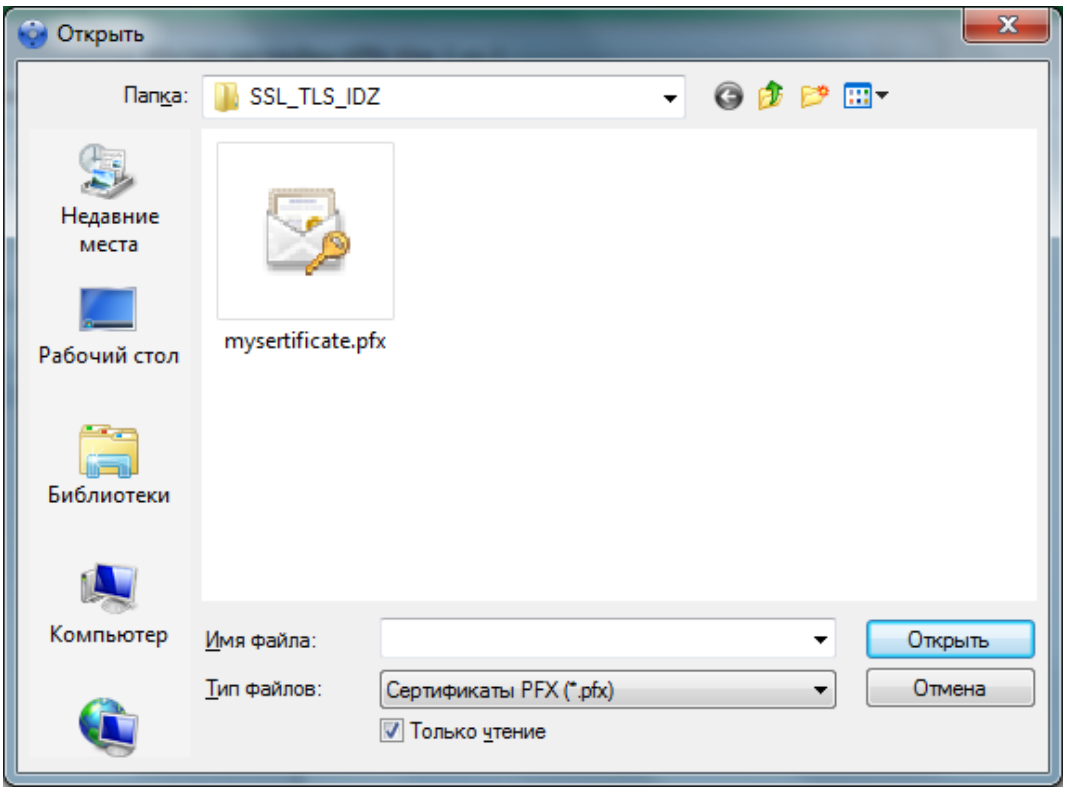


Рис. 8.11. Выбор созданного сертификата

Вводим пароль для доступа к сертификату

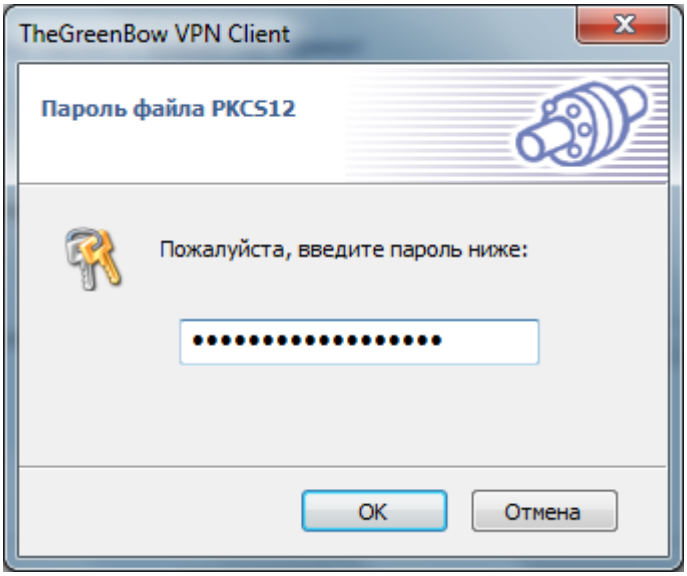


Рис. 8.12. Окно ввода пароля для доступа к сертификату

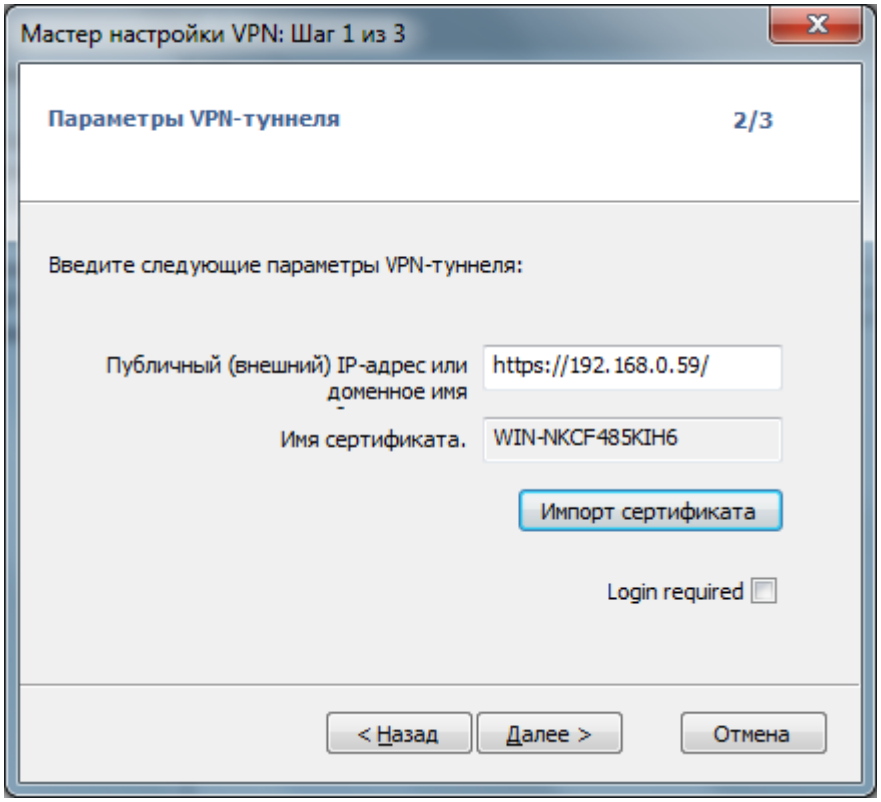


Рисунок 8.13. Окно мастера настройки VPN-туннеля

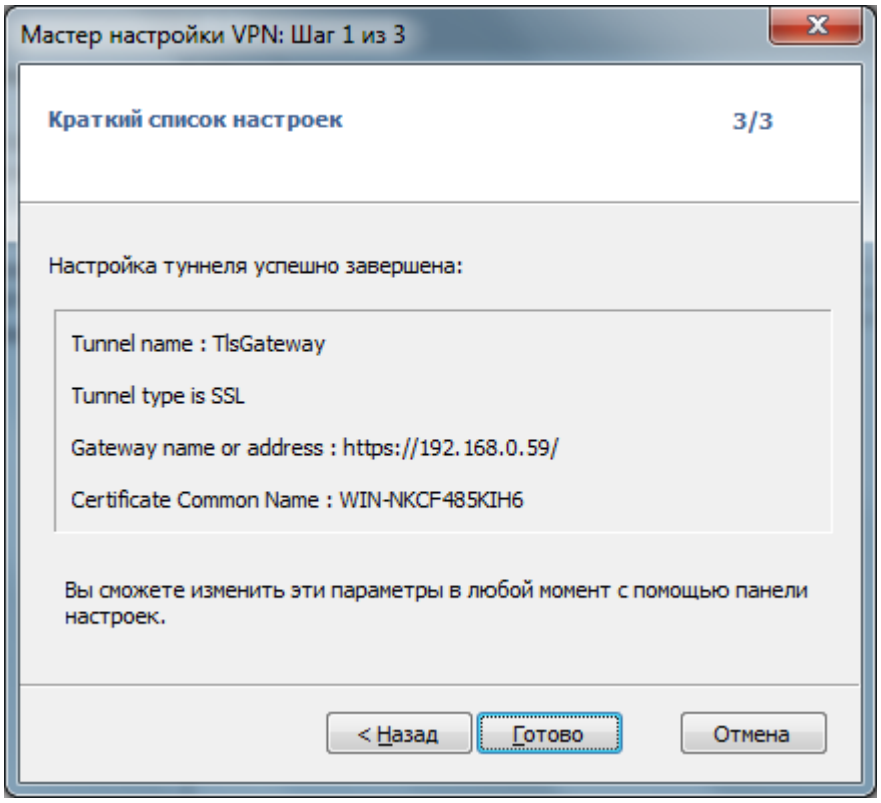


Рис. 8.14. Подтверждение об успешно созданном туннеле

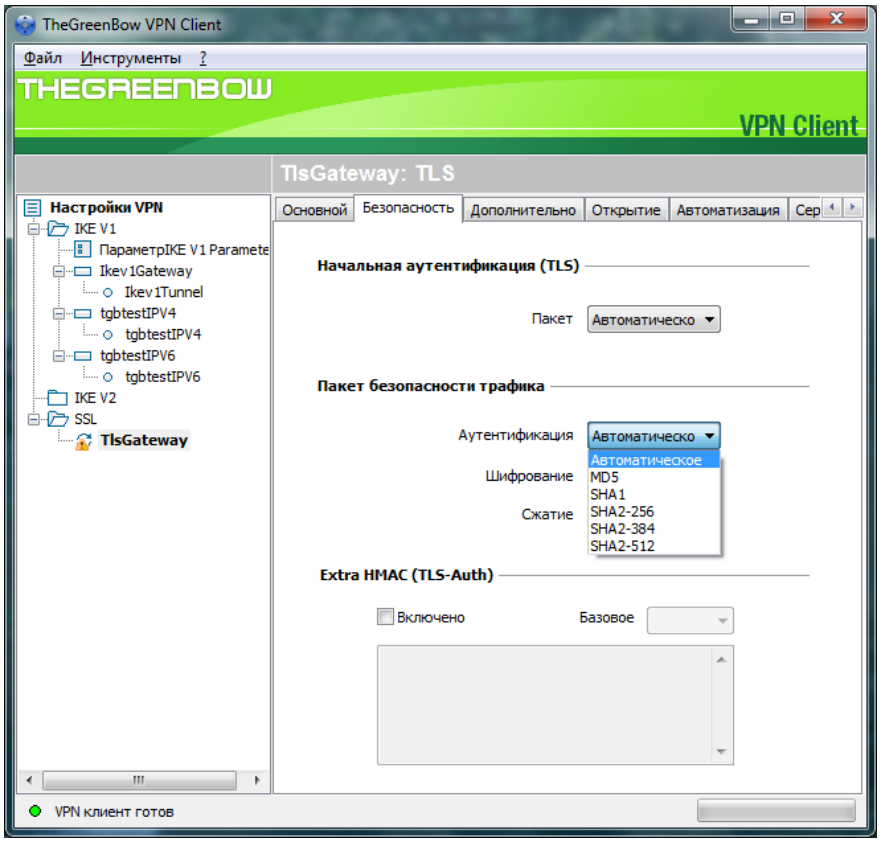


Рис. 8.15. Параметры аутентификации трафика

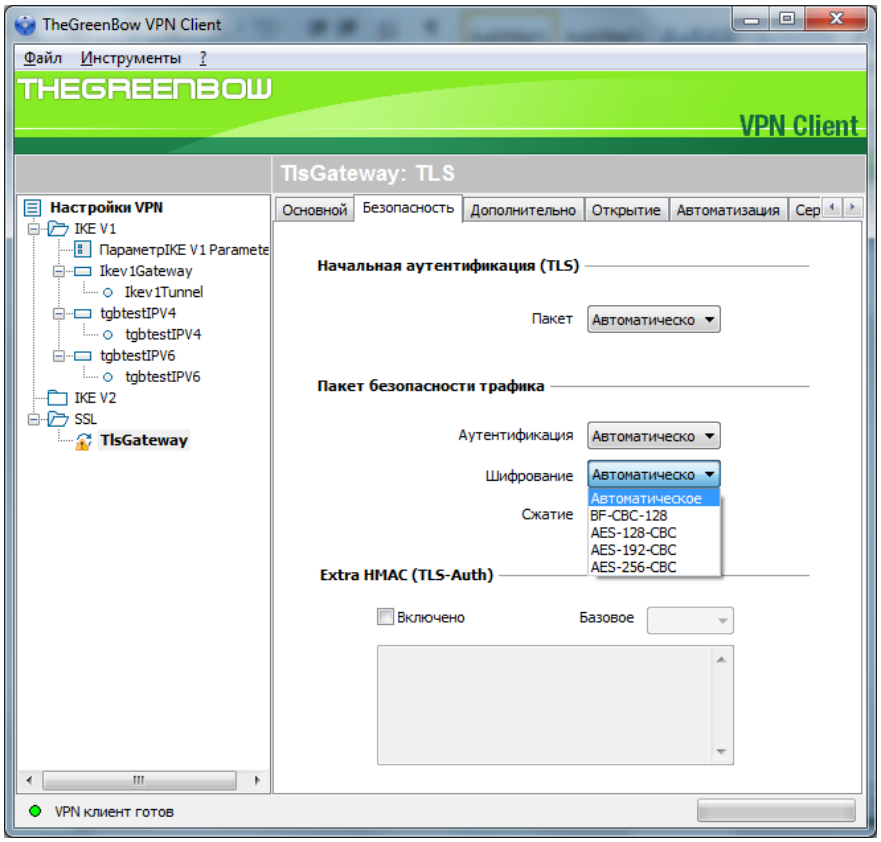


Рис. 8.16. Параметры шифрования

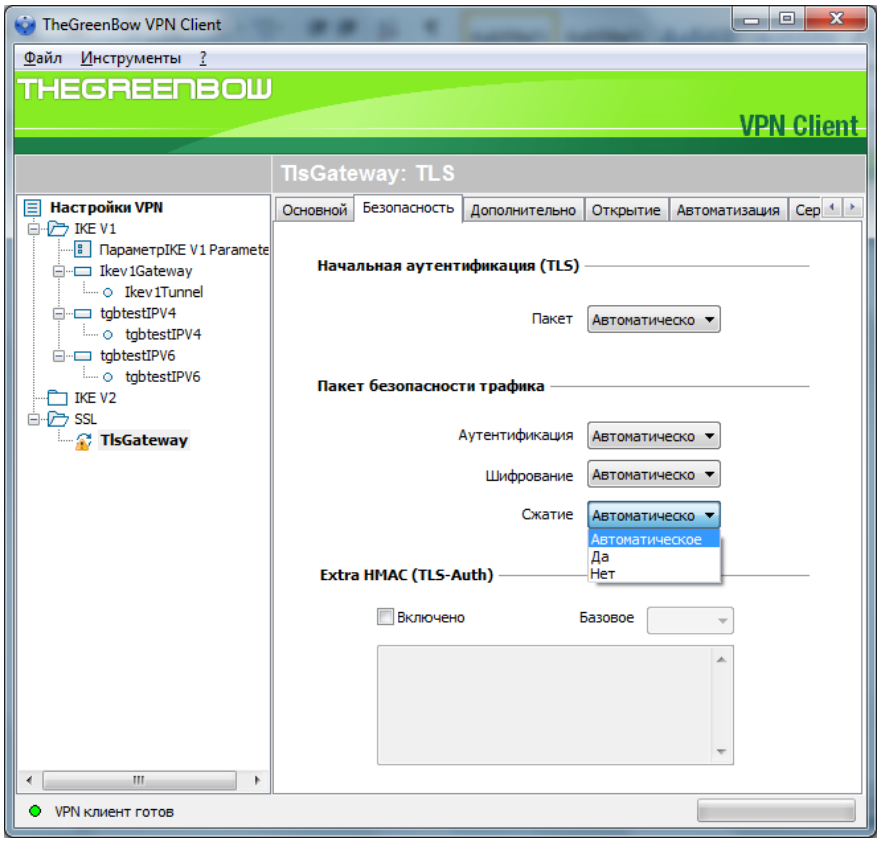


Рис.8.17. Параметры сжатия

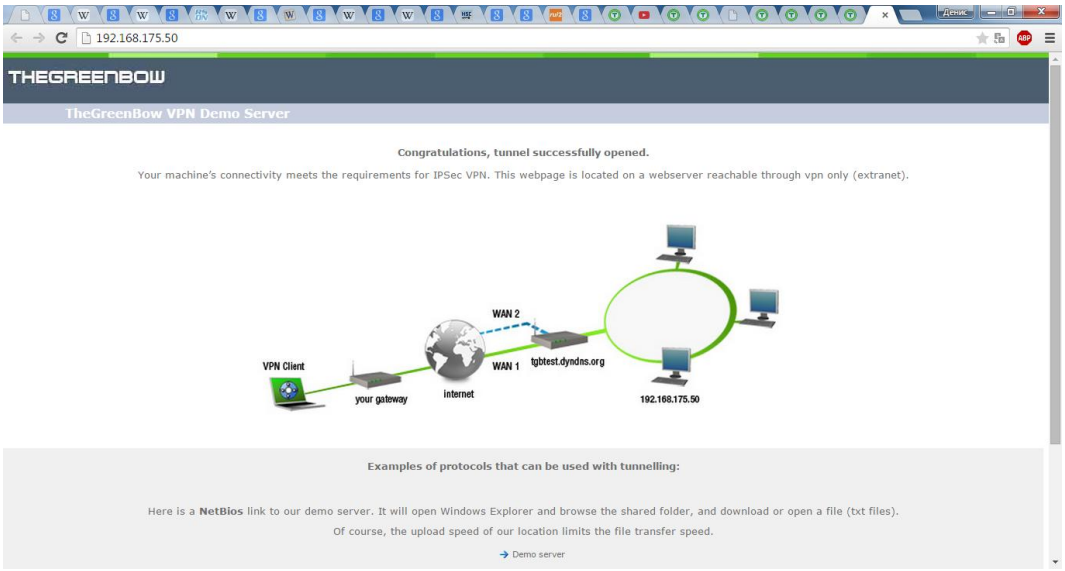


Рис. 8.18. Окно в браузере об успешном соединении

Выводы

Протокол безопасности транспортного уровня обеспечивает услуги безопасности из конца в конец для приложений, которые пользуются протоколами транспортного уровня, такими как, например, TCP. На сегодняшний день преобладает применение двух протоколов: *протокол "Уровень Безопасных Розеток" (SSL - Secure Sockets Layer)* и *протокол "Безопасность Транспортного уровня"* (TLS - Transport Layer Security).

• SSL или TLS обеспечивают такие услуги, как фрагментация, сжатие, целостность сообщения, конфиденциальность и создание кадра данных, полученных от прикладного уровня. Как правило, SSL (или TLS) может получить прикладные данные от любого протокола прикладного уровня, но работает протокол обычно с *HTTP*.

• Комбинация алгоритмов смены ключей, хэширования и алгоритм шифрования определяют *набор шифров* для каждого сеанса.

• Для того чтобы обмениваться заверенными и конфиденциальными сообщениями, клиенту и серверу необходимо иметь шесть единиц криптографической секретности (четыре ключа и два вектора инициализации).

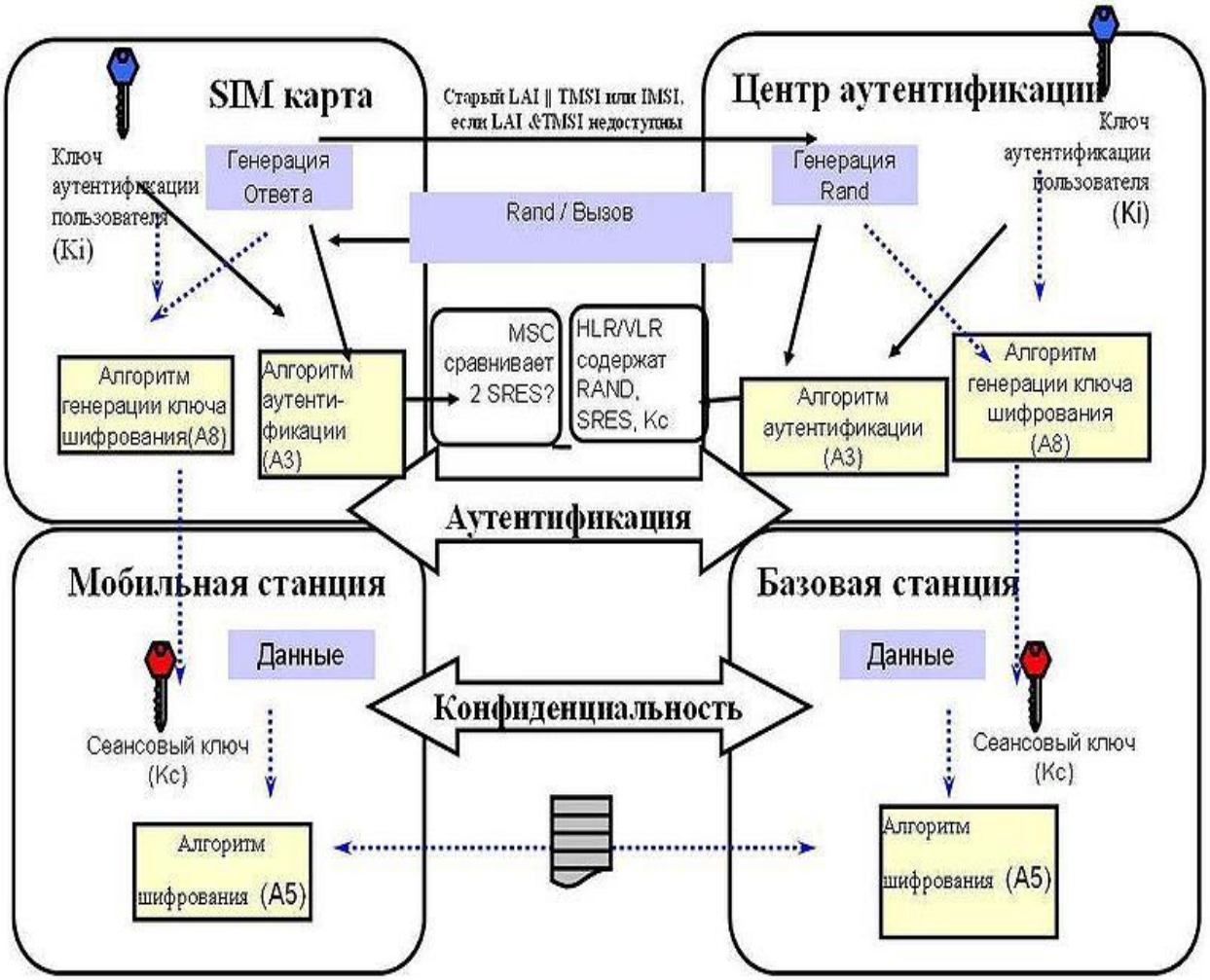
• В SSL (или TLS) отличают *подключение* и сеанс. В сеансе одна сторона играет роль клиента, а другая - роль сервера; при *подключении* обе стороны играют одинаковые роли, на равном подуровне.

• SSL (или TLS) определяет четыре протокола на двух уровнях: *протокол установления соединения*, *протокол изменения параметров шифрования, аварийный*

протокол и *протокол передачи записей*. *Протокол установления соединения* использует несколько сообщений, чтобы договориться о *наборе шифров*, подтвердить подлинность сервера для клиента и клиента для сервера, если это необходимо, и обмениваться информацией для организации криптографической секретности. *Протокол изменения параметров шифрования* определяет процесс перемещения информации между состоянием ожидания и активным состоянием. *Аварийный протокол* передает извещения об ошибках и ситуациях, отклоняющихся от нормальных. *Протокол передачи записей* доставляет сообщения от верхнего уровня (протокол установления соединения, аварийный протокол, ChangeCipherSpec-протокол) или прикладного уровня.

Лабораторная работа 9. Исследование алгоритма поточного шифрования A5/1 сотовой системы связи GSM

Прежде чем приступить к описанию алгоритма шифрования используемого в GSM сетях рассмотрим каким образом происходит аутентификация пользователя и формирования ключа шифрования. Для этого воспользуемся картинкой.



На данном рисунке схематично представлены следующие шаги:
Телефон оператора подключается к сети.

1. Для подтверждения своей подлинности телефон посылает специальный идентификационный код, называемый TMSI(Temporary Mobile Subscriber Identity).
2. Центр Аутентификации(ЦА) генерирует 128-битное случайное число RAND и посылает его на Мобильную Станцию(МС).
3. МС зашифровывает полученное число RAND, используя свой секретный ключ K_i и алгоритм аутентификации A3.

4. МС берет первые 32 бита из последовательности, полученной на предыдущем шаге(назовем их SRES(signed response)) и отправляет их обратно на ЦА.

5. ЦА проделывает ту же операцию и получает 32 битную последовательность XRES(expected response).

6. После чего ЦА сравнивает SRES и XRES. В случае, если оба значения равны, телефон считается аутентифицированным.

7. МС и ЦА вычисляют сессионный ключ шифрования, используя секретный ключ K_i и алгоритм формирования ключа A8 $K_c=A8_{ki}(RAND)$

Говоря об алгоритмах аутентификации A3 и алгоритме формирования ключа A8, следует отметить что на практике большинство сотовых операторов используют для этих целей один алгоритм, называемый COMP128(он имеет множество модификаций COMP128-1, COMP128-2, COMP128-3).
COMP128 представляет собой обыкновенную хэш-функцию, на входе которая принимает 128-битную последовательность и на выходе возвращает 96-битную.

Как всегда в криптографии, попытка сэкономить время разработчикам обернулась полным провалом. Безопасность GSM сетей изначально основывалась на принципе «безопасность за счёт неизвестности». И когда в 1998 году алгоритм был вскрыт группой исследователей состоящих из Marc Briceno, Ian Goldberg и David Wagner обранужилась одна занятная особенность: последние 10 бит секретного ключа K_i всегда равнялись нулю. Используя это любопытное свойство, а так же уязвимость COMP128 к «атаке дней рождений» Marc Briceno, Ian Goldberg и David Wagner смогли извлечь секретный ключ K_i из SIM-карты.
Результатом этого исследования стал повсеместный отказ от алгоритма COMP128 и его замена на более надежные модификации COMP128-2 и COMP128-3, технические детали которых держатся в тайне.

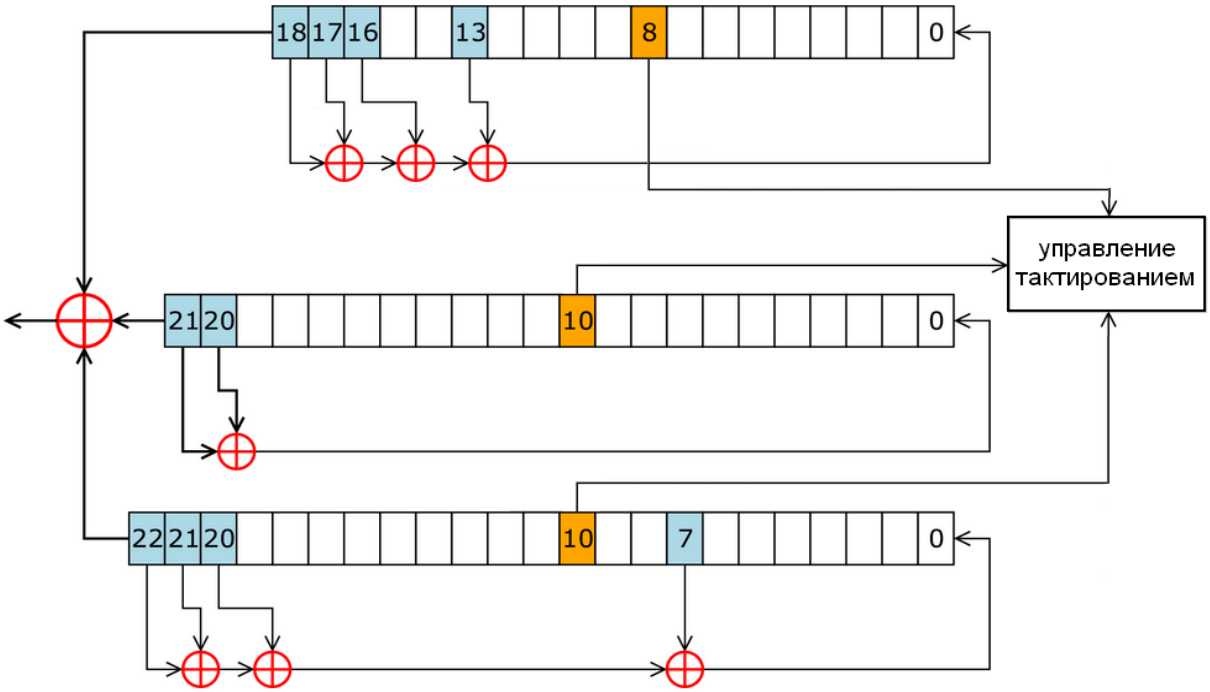
Алгоритм шифрования A5/1

В качестве алгоритма шифрования в GSM используются алгоритмы из семейства A5. На сегодняшний день их всего 3:

A5/1 — поточный шифр, наиболее распространенный на сегодня.

- **A5/2**-вариант предыдущего алгоритма «для бедных». Очень похож на своего «старшего брата», но изначально задумывался, как сильно ослабленная версия A5/1. В настоящее время не используется
- **A5/3**-блочный шифр. Разработан в 2002 году с целью заменить устаревший A5/1. Однако в настоящее время используется только в 3GPP сетях. У алгоритма найден ряд уязвимостей, но о практических атаках речи пока не идет.

Рассмотрим подробнее алгоритм A5/1.



Внутреннее состояние шифра A5/1 состоит из трех линейных регистров сдвига с обратной связью R1, R2, R3, длиной 19, 22 и 23 бита соответственно(всего 64 бита).

Сдвиг в регистрах R1, R2, R3 происходит только при выполнении определенного условия. Каждый регистр содержит " бит управления тактированием". В R1 это 8-й бит, а в R2 и R3 — 10-й. На каждом шаге сдвигаются только те регистры у которых значение бита синхронизации равно большинству значений синхронизирующих битов всех трех регистров.

На сегодняшний день известно большое количество успешных атак на GSM шифрование и все они относятся к атакам типа known-plaintext, т.е. для восстановления ключа атакующему помимо зашифрованных фреймов необходимо знать так же незашифрованные данные, которые соответствуют этим фреймам. На первый взгляд такое требование может показаться фантастическим, однако из-за специфики стандарта GSM, в котором помимо голосового трафика передаются различные системные сообщения, такого рода атаки из разряда теоретических переходят в разряд практических.

Системные сообщения GSM содержат повторяющиеся данные и могут использоваться злоумышленником. В частности метод, предложенный Karsten Nohl в 2010 году основан как раз таки на поиске такого рода данных в шифротексте и простом переборе различных вариантов ключей, хранящихся в радужных таблицах, до тех пор пока не будет найден ключ, порождающий нужный шифротекст для известного заранее системного сообщения.

<u>Лабораторная работа 10.</u> Криптографическая защита беспроводной сети LTE

Стандарт сетей LTE [16] – стандарт беспроводной высокоскоростной передачи данных для мобильных телефонов и других терминалов, работающих с данными. Он основан на GSM/EDGE и UMTS/HSPA сетевых технологиях, увеличивая пропускную способность и скорость за счёт использования другого радиоинтерфейса вместе с улучшением ядра сети.

На рисунке 5.1 представлена структура сети стандарта LTE.

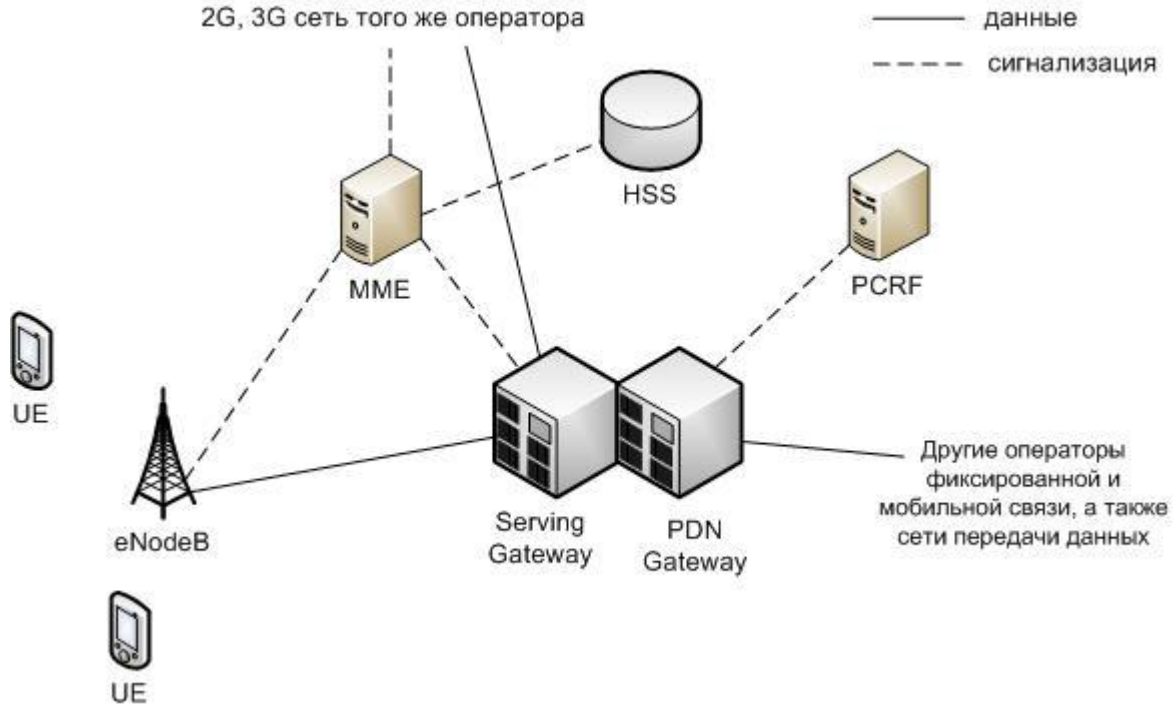


Рис. 10.1. Структура сети стандарта LTE

Из этой схемы видно, что структура сети сильно отличается от сетей стандартов 2G и 3G. Существенные изменения претерпела и подсистема базовых станций, и подсистема коммутации. Изменена технология передачи данных между оборудованием пользователя и базовой станцией. Также подверглись изменению и протоколы передачи данных между сетевыми элементами. Вся информация (голос, данные) передается в виде пакетов. Таким образом, уже нет разделения на части обрабатывающие либо только голосовую информацию, либо только пакетные данные.

Можно выделить следующие основные элементы сети стандарта LTE:

- **Serving SAE Gateway** или просто **Serving Gateway** *(SGW)* – обслуживающий шлюз сети LTE. Предназначен для обработки и маршрутизации пакетных данных поступающих из/в подсистему базовых станций. SGW имеет прямое соединение с сетями

второго и третьего поколений того же оператора, что упрощает передачу соединения в /из них по причинам ухудшения зоны покрытия, перегрузок и т.п. В SGW нет функции коммутации каналов для голосовых соединений, т.к. в LTE вся информация, включая голос коммутируется и передается с помощью пакетов.

- **Public Data Network SAE Gateway** или просто **PDN Gateway** *(PGW)* – шлюз к сетям передачи данных других операторов для сети LTE. Основная задача PGW заключается в маршрутизации трафика сети LTE к другим сетям передачи данных, таких как Интернет, а также сетям GSM, UMTS.
- **Mobility Management Entity** *(MME)* – узел управления мобильностью сети сотовой связи стандарта LTE. Предназначен для обработки сигнализации, преимущественно связанной с управлением мобильностью абонентов в сети.
- **Home Subscriber Server** *(HSS)* – сервер абонентских данных сети сотовой связи стандарта LTE. Представляет собой большую базу данных и предназначен для хранения данных об абонентах. Кроме того, HSS генерирует данные, необходимые для осуществления процедур шифрования, аутентификации и т.п. Сеть LTE может включать один или несколько HSS. Количество HSS зависит от географической структуры сети и числа абонентов.
- **Policy and Charging Rules Function** *(PCRF)* – элемент сети сотовой связи стандарта LTE, отвечающий за управление начислением платы за оказанные услуги связи, а также за качество соединений в соответствии с заданными конкретному абоненту характеристиками.

Для того чтобы данные могли быть транспортированы через интерфейс радио LTE, используются различные «каналы». Они используются для того, чтобы выделять различные типы данных и позволить им транспортироваться через сеть доступа более эффективно. Использование нескольких каналов обеспечивает интерфейс более высокого уровня в рамках протокола LTE и включают более чёткую и определенную сегрегацию данных.

Есть три категории, в которые могут быть сгруппированы различные каналы передачи данных:

Логические каналы – предоставляет услуги среднего уровня управления доступом MAC *(Medium Access Control)* в пределах структуры протокола LTE. Логические каналы по типу передаваемой информации делятся на логические каналы управления и логические каналы трафика. Логические каналы управления используются для передачи различных сигнальных и информационных сообщений. По логическим каналам трафика передают пользовательские данные.

Транспортные каналы — транспортные каналы физического уровня предлагают передачу информации в MAC и выше. Информацию логических каналов после обработки на RLC/MAC уровнях размещают в транспортных каналах для дальнейшей передачи по радиоинтерфейсу в физических каналах. Транспортный канал определяет как и с какими характеристиками происходит передача информации по радиоинтерфейсу. Информационные сообщения на транспортном уровне разбивают на транспортные блоки. В каждом временном интервале передачи (Transmission Time Interval, TTI) по радиоинтерфейсу передают хотя бы один транспортный блок. При использовании технологии MIMO возможна передача до четырех блоков в одном TTI.

Физические каналы – это каналы передачи, которые переносят пользовательские данные и управляющие сообщения. Они изменяются между восходящим и нисходящим потоками, поскольку каждый из них имеет различные требования и действует по-своему.

Существующие методы и стандарты защиты беспроводных сетей LTE

Безопасность в сетях LTE заключается в нескольких видах:

- Защита абонентов.
- Защита передаваемых сообщений.
- Шифрование сообщений.
- Аутентификация и абонента, и сети.

Защита абонента заключается в том, что в процессе обслуживания его скрывают временными идентификаторами.

Для закрытия данных в сетях LTE используется потоковое шифрование методом наложения на открытую информацию псевдослучайной последовательности (ПСП) с помощью оператора XOR (исключающее или). В этих сетях для обеспечения безопасности внутри сети применяется принцип туннелирования соединений. Шифрации можно подвергать пакеты S1 и X2 при помощи IPsec ESP, а также подвергаются шифрации сигнальные сообщения этих интерфейсов.

В момент подключения или активизации абонентского оборудования (UE) в сети, сеть запускает процедуру аутентификации и соглашения о ключах AKA (Authentication and Key Agreement). Целью этой процедуры является взаимная аутентификация абонента и сети и выработка промежуточного ключа K_{ASME}. Работа механизма AKA занимает доли секунды, которые необходимы для выработки ключа в приложении USIM и для установления соединения с Центром регистрации (HSS). Вследствие этого, для достижения скорости передачи данных сетей LTE необходимо добавить функцию обновления ключевой информации без инициализации механизма AKA. Для решения этой проблемы в сетях LTE предлагается использовать иерархическую ключевую инфраструктуру. Здесь также, как и в

сетях 3G, приложение USIM и Центр аутентификации (AuC) осуществляет предварительное распределение ключей. Когда механизм AKA инициализируется для осуществления двусторонней аутентификации пользователя и сети, генерируются ключ шифрования CK и ключ общей защиты, которые затем передаются из ПО USIM в Мобильное оборудование (ME) и из Центра аутентификации в Центр регистрации (HSS). ME и HSS, используя ключевую пару (CK;IK) и ID используемой сети, вырабатывает ключ K_{ASME}. Установив зависимость ключа от ID сети, Центр регистрации гарантирует возможность использования ключа только в рамках этой сети. Далее K_{ASME} передается из Центра регистрации в устройство мобильного управления (MME) текущей сети, где он используется в качестве мастер-ключа. На основании K_{ASME} вырабатывается ключ $K_{nas\text{-}enc}$, который необходим для шифрования данных протокола NAS между мобильным устройством (UE) и MME, и $K_{nas\text{-}int}$, необходимый для защиты целостности. Когда UE подключается к сети, MME генерирует ключ KeNB и передает его базовым станциям. В свою очередь, из ключа KeNB вырабатывается ключ Kup-enc, используемый для шифрования пользовательских данных протокола U-Plane, ключ Krrc-enc для протокола RRC (Radio Resource Control - протокол взаимодействия между Мобильными устройствами и базовыми станциями) и ключ Krrc-int, предназначенный для защиты целостности.

Алгоритм аутентификации и генерации ключа представлен на рис 10.2

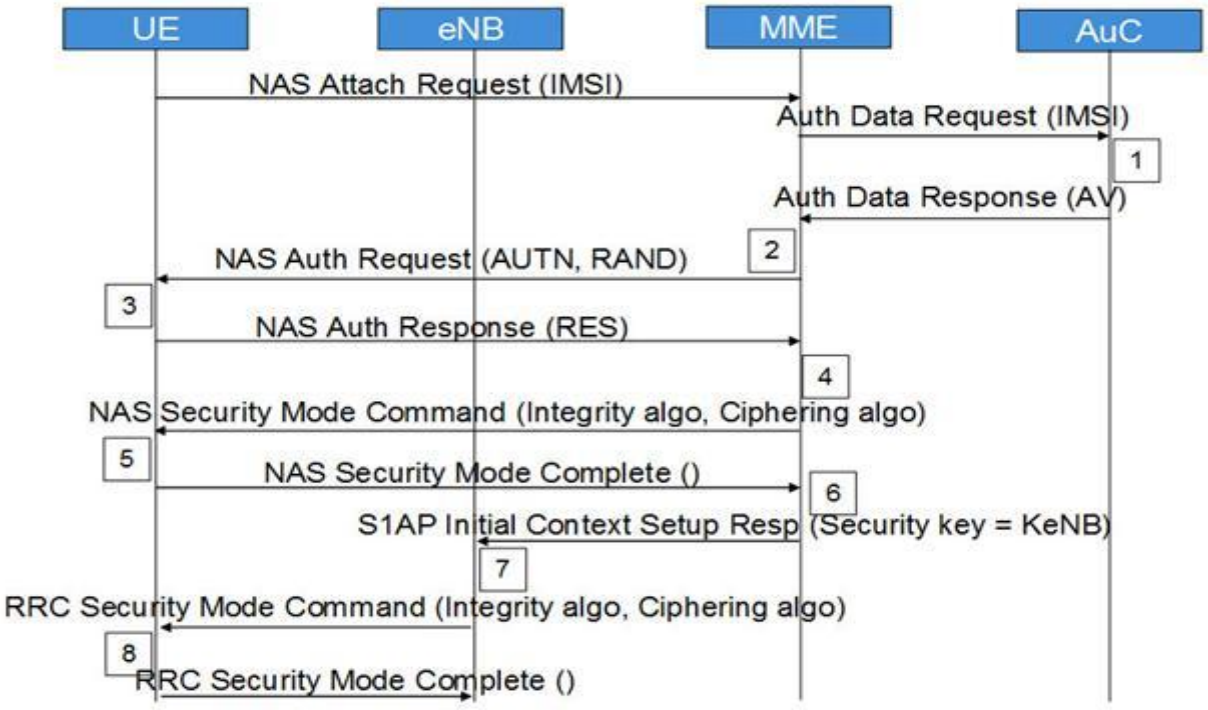


Рис. 10.2. Алгоритм аутентификации и генерации ключа

Здесь:

Шаг 1. Запрос о подключении к сети от мобильной станции (UE). MME запрашивает аутентификационные данные, относящиеся к конкретному IMSI, отправляя Authentication Data Request. AuC/HSS выбирает PSK, относящийся к конкретному IMSI и вычисляет аутентификационные данные по PSK. AuC/HSS отправляет обратно AV с Authentication Data Response.

Шаг 2. MME получает IK, CK, XRES, RAND и AUTH из AV. MME отправляет AUTH и RAND при помощи Authentication Request к UE.

Шаг 3. UE аутентифицирует NW, проверяя полученный AUTH. После чего вычисляет IK, CK, RES, XMAC из своего ключа защиты, AMF, (OP), AUTH и RAND. Она отправляет RES с Authentication response.

Шаг 4. После получения RES, MME сравнивает его с XRES и если они совпадают, то аутентификация прошла успешно, в противном случае, MME отправляет сбой аутентификации (Authentication failure) к UE. MME сбрасывает счетчик DL NAS. Рассчитывает KASME, KeNB, Knas-int, Knas-enc. Отправляет NAS команду режима безопасности (алгоритм целостности, алгоритм шифрования, NAS набор ключей ID, функцию безопасности UE) с целостностью охраняемых, но не зашифрованных, используя Knas-inc.

Шаг 5. После получения NAS команды режима безопасности, UE вычисляет KASME, KeNB, Knas-int, Knas-enc. UE отправляет NAS режима безопасности выполнен с целостностью, защищенных и зашифрованных.

Шаг 6. После получения NAS команды режима безопасности от UE, MME отправляет KeNB в eNB с S1AP первоначальная установка начального контекста (ключ защиты).

Шаг 7. После получения KeNB, eNB вычисляет Krrc-int, Krrc-enc, Kup-enc. Затем оно отправляет RRC ключ защиты команду с AS целостностью алгоритма и AS шифрующий алгоритм.

Шаг 8. После получения RRC команды ключа защиты UE вычисляет Krrc-int, Krrc-enc, Kup-enc. UE отправляет RRC выполненный ключ шифрования на eNB.

После всех описанных действий, все NAS и AS сообщения будут надежно защищены и зашифрованы, в отличие от пользовательских данных, которые будут только шифроваться.

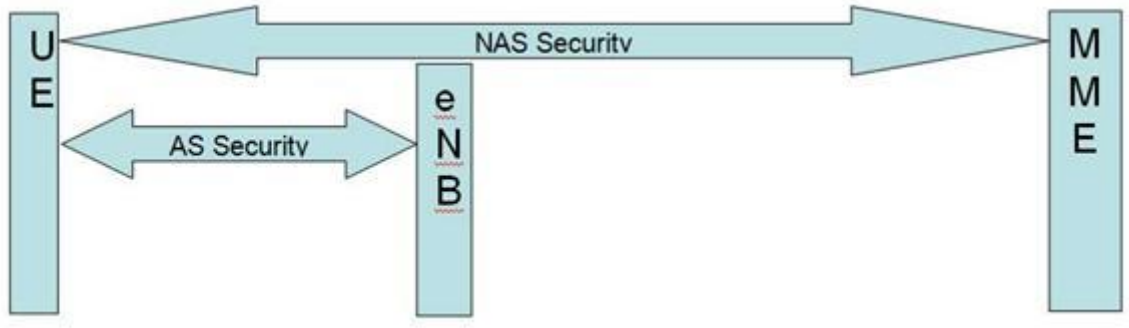


Рис. 10.3. Слои безопасности

Архитектура безопасности LTE определяет механизм безопасности и для уровня NAS и для уровня AS.

Безопасность NAS (слоя без доступа): Выполнена для NAS сообщений и принадлежит области UE и MME.

В этом случае необходима при передаче сообщений NAS между UE и MME – целостность, защищенная и зашифрованная с дополнительным заголовком безопасности NAS.

Безопасность AS (слоя с доступом): Выполнена для RRC и плоскости пользовательских данных, принадлежащих области UE и eNB. Уровень PDCP на сторонах UE и eNB отвечает за шифрование и защиту целостности.

RRC сообщения защищены целостностью и зашифрованы, однако данные U-Plane только зашифрованы.

Для генерации векторов аутентификации используется криптографический алгоритм с помощью однонаправленных функций (f1, f2, f3, f4, f5) когда прямой результат получается путем простых вычислений, а обратный результат не может быть получен обратным путем, то есть не существует эффективного алгоритма получения обратного результата. Для этого алгоритма используется случайное 128 битное случайное число RAND, мастер-ключ K абонента, также 128 бит и порядковый номер процедуры SQN (Sequence Number). Счетчик SQN меняет свое значение при каждой генерации вектора аутентификации. Похожий счетчик SQN работает и в USIM. Такой метод позволяет генерировать каждый раз новый вектор аутентификации, не повторяя предыдущий уже использованный вектор аутентификации.

Помимо этих трех исходных величин: SQN, RAND и K в алгоритме f1 участвует поле управления аутентификацией Authentication Management Field (AMF), а в алгоритмах f2 – f5 исходные параметры – RAND и K, что и продемонстрировано на рис. 2.3, 2.4. На выходах соответствующих функций получают Message Authentication Code (MAC) - 64 бита; XRES – eXpected Response, результат работы алгоритма аутентификации <32 – 128 бит>; ключ

шифрации CK, генерируемый с использованием входящих (K,RAND)->f3->CK; ключ целостности IK, сгенерированный с использованием входящего (K,RAND)->f4->IK; и промежуточный ключ Anonymity Key (AK), генерируемый с помощью (K,RAND)->f5->AK - 64 бита.

При обслуживании абонента сетью E-UTRAN ключи CK и IK в открытом виде в ядро сети не передают. В этом случае HSS генерирует K_{ASME} с помощью алгоритма KDF (Key Derivation Function), для которого исходными параметрами являются CK и IK, а также идентификатор обслуживающей сети и SQNÅAK. Вектор аутентификации содержит RAND, XRES, AUTN и K_{ASME}, на основе которого происходит генерация ключей шифрации и целостности, используемых в соответствующих алгоритмах.

Когда мобильная станция получает из ядра сети три параметра (RAND, AUTN и KSI_{ASME}, где KSI – Key Set Identifier, индикатор установленного ключа, однозначно связанный с K_{ASME} в мобильной станции).

После чего используя RAND и AUTN, USIM на основе алгоритмов безопасности, тождественных хранящимся в HSS, производит вычисление XMAC, RES, CK и IK.

Затем в ответе RES UE передает в MME вычисленное RES, которое должно совпасть с XRES, полученным из HSS. Так сеть аутентифицирует абонента. Вычислив XMAC, UE сравнивает его с MAC, полученным ею в AUTN. При успешной аутентификации абонентом сети (MAC = XMAC) UE сообщает об этом в ответе RES. Если аутентификация сети не удалась (MAC ≠ XMAC), то UE направляет в MME ответ CAUSE, где указывает причину неудачи аутентификации.

При успешном завершении предыдущего этапа MME, eNB и UE производят генерацию ключей, используемых для шифрации и проверки целостности получаемых сообщений. В E-UTRAN имеется иерархия ключей, которая приведена на рис. 2.5.

Векторы аутентификации (рис. 2.3, 2.4):

Ключи IK и CK генерируются и в центре аутентификации, и в USIM;

Ключ AK генерируется только в центре аутентификации;

Ответ XRES генерируется только в центре аутентификации, а RES генерируется в USIM;

Код MAC генерируется только в центре аутентификации, а соответствующий ему параметр XMAC генерируется в USIM;

Маркер AUTH генерируется только в центре аутентификации.

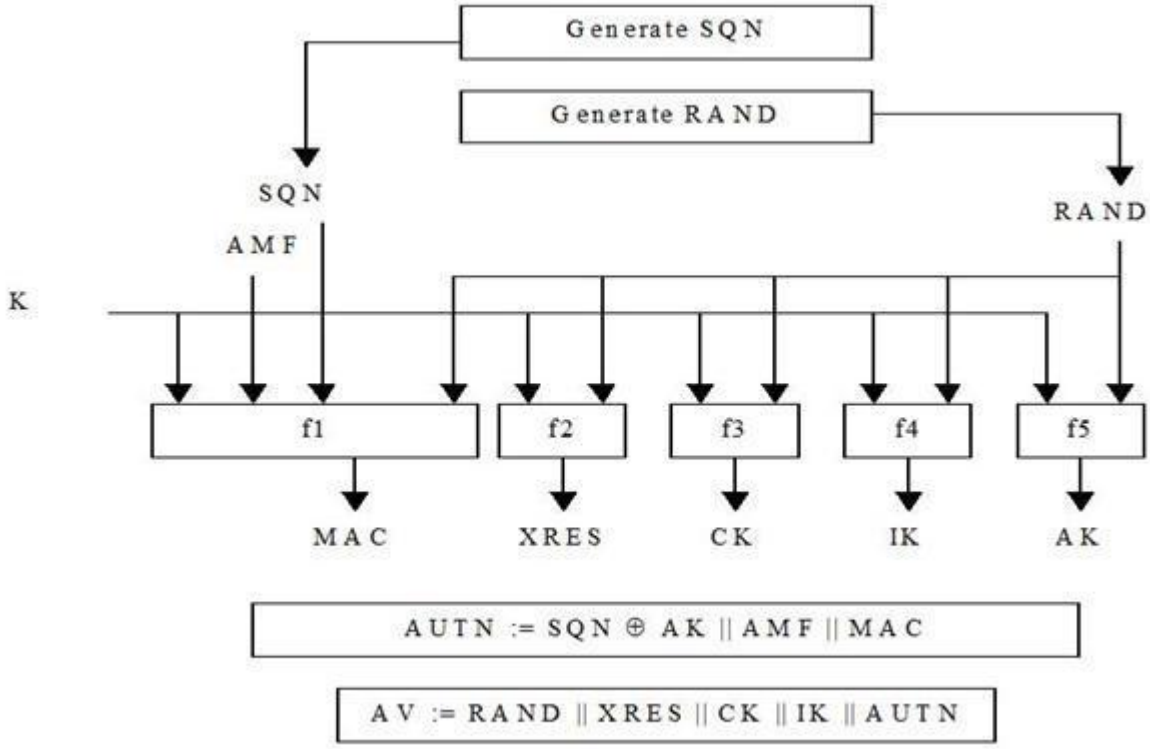


Рис. 10.4. Создание векторов на передающей стороне

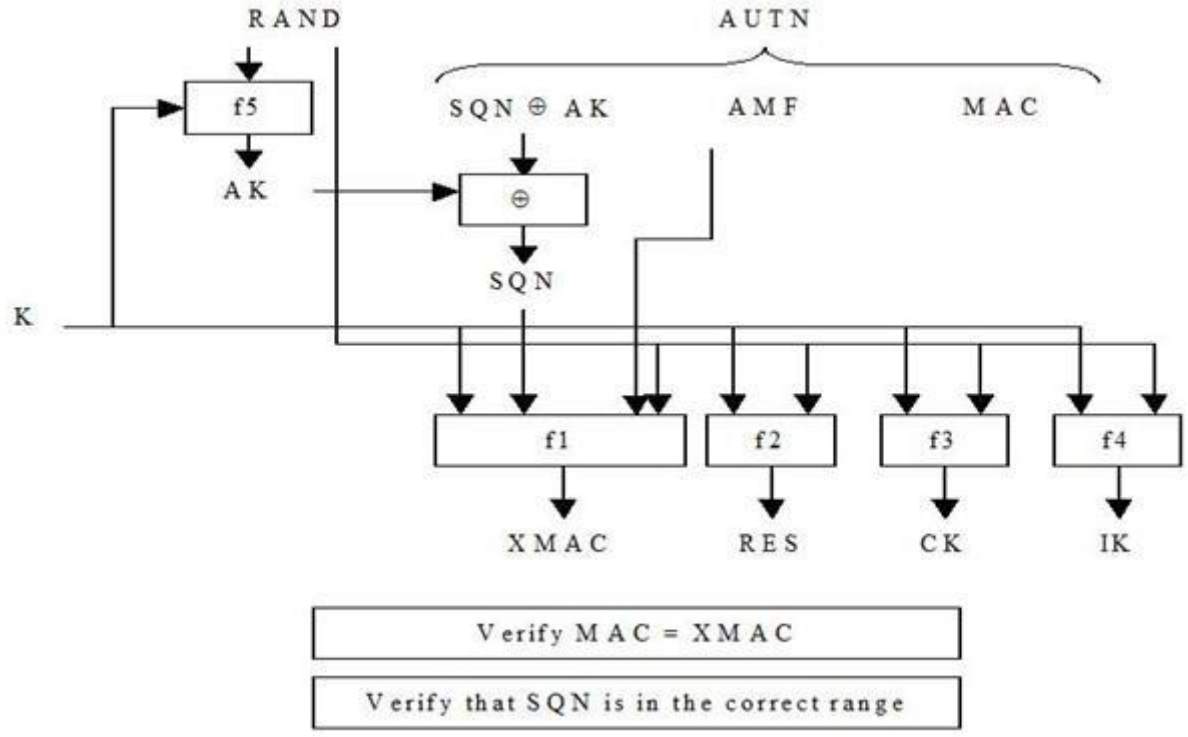


Рис. 10.5. Преобразование векторов на принимаемой стороне

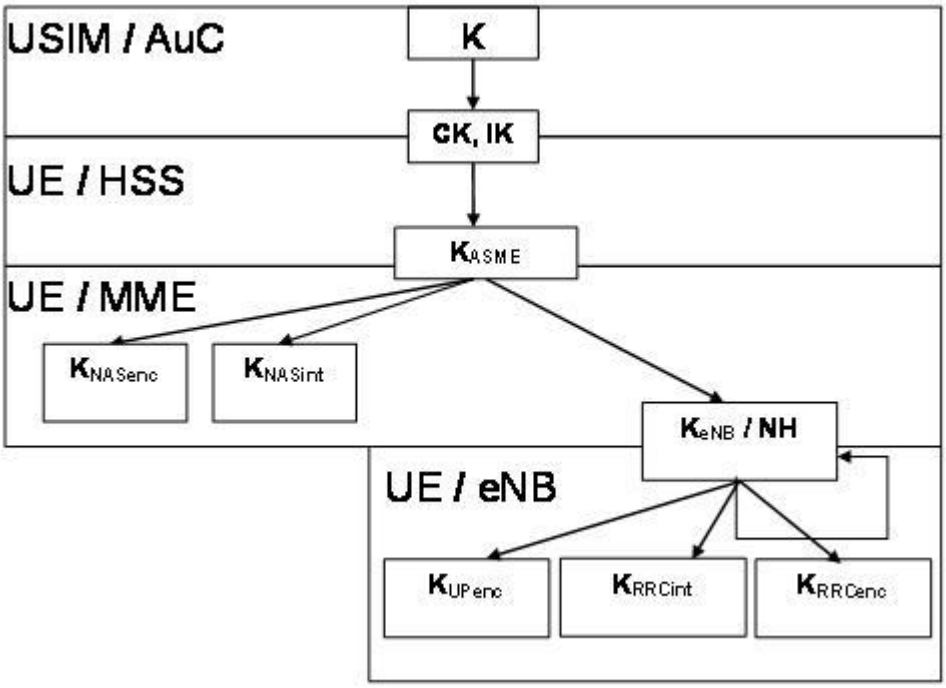


Рис. 10.6. Иерархия ключей в E-UTRAN

Исходным ключом для всей цепочки является K_{ASME} (256 бит). При передаче в радиоканале защиту обеспечивают для сигнального трафика (Control Plane) и для пользовательских пакетов (User Plane). При этом все сообщения сигнализации разделяют на сквозные сигнальные сообщения между UE и MME протоколов MM и SM (NAS – Non Access Stratum) и сигнальные сообщения между eNB протокола RRC (AS – Access Stratum). Для шифрации и защиты целостности можно использовать разные базовые алгоритмы:

- UEA2 (UMTS Encryption Algorithm 2) и UIA2 (UMTS Integrity Algorithm 2);
- разработанные для стандартов 3G,AES (Advanced Encryption Standard).

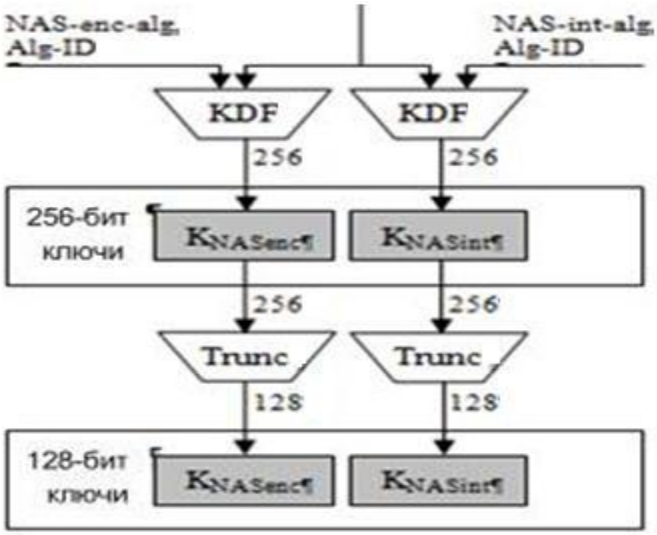


Рис. 10.7. Генерирование ключей шифрации и целостности для NAS сигнализации

Сигнальные сообщения протокола RRC (AS) также шифруют и обеспечивают их целостность. Пакеты трафика только шифруют. Эти операции производят в обслуживающей eNB и UE. Схема получения ключей шифрации и целостности (рис. 7) для AS и UP трафика отличается от предыдущего случая тем, что исходным параметром здесь служит вторичный промежуточный ключ KeNB (256 бит). Этот ключ генерируют, также используя KDF, где входными параметрами являются: KASME, счетчик сигнальных сообщений NAS вверх, прежнее значение KeNB, идентификатор соты и номер частотного канала в направлении вверх. Следовательно, при каждой периодической локализации UE происходит изменение KeNB.

Также KeNB меняется и при хэндовере; при этом в алгоритме генерации нового KeNB можно использовать дополнительный параметр NH (Next Hop), фактически счетчик числа базовых станций, по цепочке обслуживающих абонента. Все реализуемые процедуры безопасности в сети E-UTRAN продемонстрированы на рис. 2.8.

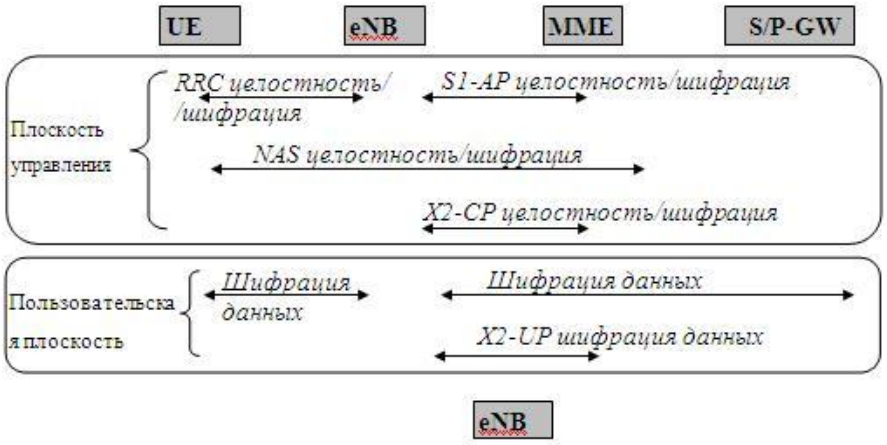


Рис. 10.8. Реализуемые процедуры безопасности в сети E-UTRAN

Алгоритм шифрации и дешифрации сообщений представлен на рис. 2.9.

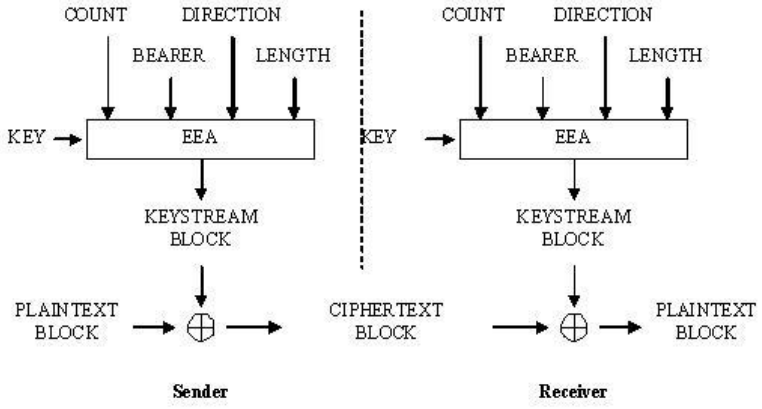


Рис. 10.9. Алгоритм шифрации в E-UTRAN

Исходными параметрами в этом алгоритме являются шифрующий ключ KEY (128 бит), счетчик пакетов (блоков) COUNT (32 бита), идентификатор сквозного канала BEARER

(5 бит), указатель направления передачи DIRECTION (1 бит) и длина шифрующего ключа LENGTH. В соответствии с выбранным алгоритмом шифрации EEA (EPS Encryption Algorithm) вырабатывается шифрующее число KEYSTREAM BLOCK, которое при передаче складывают по модулю два с шифруемым исходным текстом блока PLAINTEXT BLOCK. При дешифрации на приемном конце повторно совершают эту же операцию.

Процедура защиты целостности сообщения состоит в генерации “хвоста“ MAC (Message Authentication Code) (32 бита), присоединяемого к передаваемому пакету. Алгоритм генерации MAC и проверки целостности полученного пакета путем сравнения XMAC с MAC (они должны совпасть) отображен на рис. 5.10.

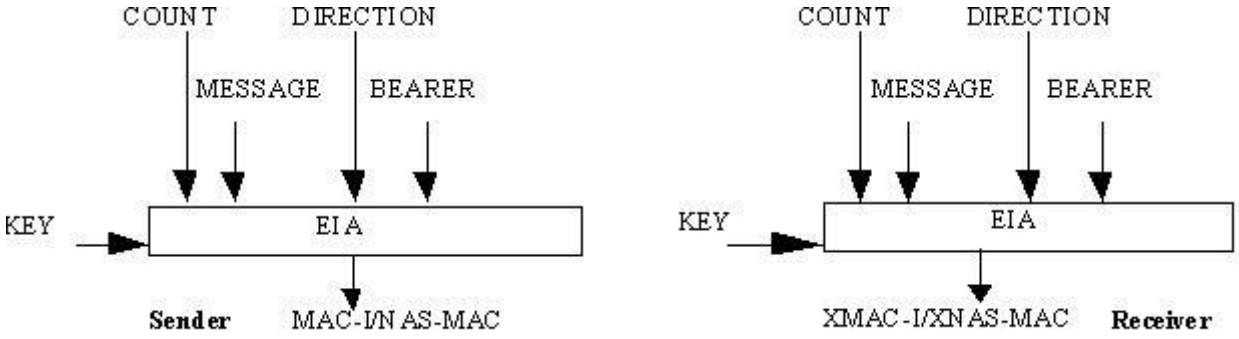


Рис. 10.10. Алгоритм проверки целостности E-UTRAN

В алгоритме EIA (EPS Integrity Algorithm) использован ключ целостности KEY (128 бит), счетчик сообщений COUNT (32 бита), идентификатор сквозного канала BEARER (5 бит), указатель направления передачи DIRECTION (1 бит) и само сообщение MESSAGE.

Моделирование технологии LTE в среде MATLAB с использованием встроенного пакета LTE System Toolbox

LTE System Toolbox™ предоставляет соответствующие стандарту функции и приложения для проектирования, моделирования и проверки коммуникационных систем стандартов LTE и LTE-Advanced. Данный инструмент ускоряет разработку LTE-алгоритмов и физического уровня (PHY), предоставляет эталонный образец для проверки и тестирования на соответ- ствие стандарту, а также позволяет генерировать тестовые сигналы. С помощью LTE System Toolbox можно производить настройку, моделирование, измерения и анализ канала связи. Также можно создавать и повторно использовать сценарии тестов для подтверждения того, что проекты, прототипы и разработки соответствуют стандарту LTE.

Основные особенности:

• Модели, соответствующие стандартам LTE и LTE-Advanced (Release 8, 9, 10 и 11). • Функции обработки на канальном уровне, поддержка от 1 до 10 режимов передачи по нисходящему каналу, образцы проектов, в том числе CoMP.

• Тестовые модели (E-TM), эталонный измерительный канал (RMC) для LTE, LTE-A, а также генератор UMTS-сигналов.

• Интерактивные инструменты для проверки на соответствие стандарту и BER тестов.

• Передача и приём сигналов при помощи радиоустройств для тестирования систем в реаль- ном эфире.

• Выделение системных и контрольных параметров из принятого сигнала, в том числе cell ID, MIB и SIB1. • Оценка канала связи.

Сквозное моделирование LTE

System Toolbox даёт возможность моделировать и имитировать физический уровень стандарта LTE. Моделирование системы на канальном уровне позволяет добиться требуемых значений характеристик системы, в том числе пропускной способности и BER, а также опреде- лить конкретные реализации системы на основе производимых измерений.

LTE System Toolbox также позволяет улучшить планирование системы, облегчая моделирование канального уровня, которое предоставляет некоторые параметры, необходимые для проектирования базовых станций с заданной геометрией и характеристиками распространения сигнала.

Набор поддерживаемых функций для моделирования режимов передачи и приёма, а также канала связи, включает в себя:

• режимы FDD и TDD на несущих частотах; • все полосы передачи LTE-сигналов от 1,4 до 20 МГц, LTE-A до 100 МГц с агрегацией несущей; • различные типы LTE-сигналов, включая нисходящие и восходящие опорные сигналы и сигналы синхронизации;

• физические LTE-каналы, в том числе каналы управления и каналы общего доступа; • готовую процедуру обработки нисходящего канала, в том числе формирование нисходящего общего канала и канала управления, все возможные MIMO-режимы и генерацию OFDM-сигналов;

• готовую процедуру обработки восходящего канала, в том числе формирование восходящих общего канала и канала управления, SU-MIMO и MU-MIMO режимы и генерацию SC-FDMA сигналов;

• адаптацию к каналу связи, включая схемы выбора типов модуляции и кодирования (MCS) в соответствии с оценкой качества канала связи (CQI), индикатора ранга (RI) и индикации матрицы прекодера (PMI);

• возможности и примеры построения LTE-Advanced, в том числе приём и передача с несколькими eNB (CoMP) и с агрегацией несущей;

• модели распространения LTE-сигналов, в том числе модель пешехода (EPA), модель автомобиля (EVA), модель типичный городской застройки (ETU), модель распространения в движении, а также модели MIMO-каналов в скоростном поезде. LTE System Toolbox даёт возможность создавать тесты, измеряющие пропускную способность PDSCH-канала в соответствии с указанными в стандарте LTE (TS 36.101) условиями испытаний. Структуры данных в LTE System Toolbox позволяют удобно отображать все параметры системы. Функции данного инструмента отражают любые возможные комбинации режимов работы передатчиков, моделей каналов и приемников. Используя этот инструмент для тестирования на соответствие стандарту и BLER-тестирования, вы можете измерять характеристики системы и сравнивать их с указанными в спецификации к стандарту

На рисунке 10.12 изображена структурная схема программного комплекса.

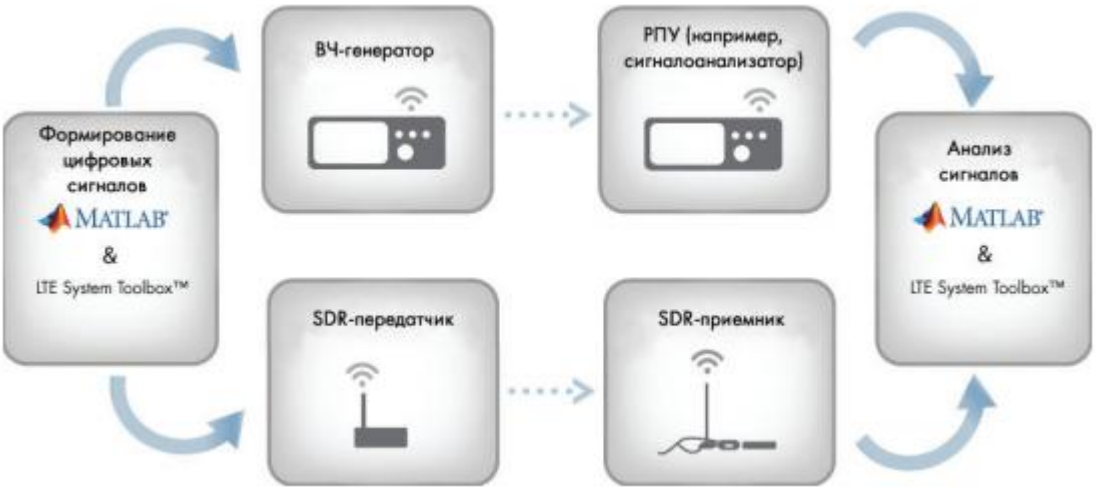


Рис. 10.12. Структурная схема программного комплекса

Для генерации тестового сигнала от базовой станции к абоненту используется генератор LTE-Downlink E-TM Generator.

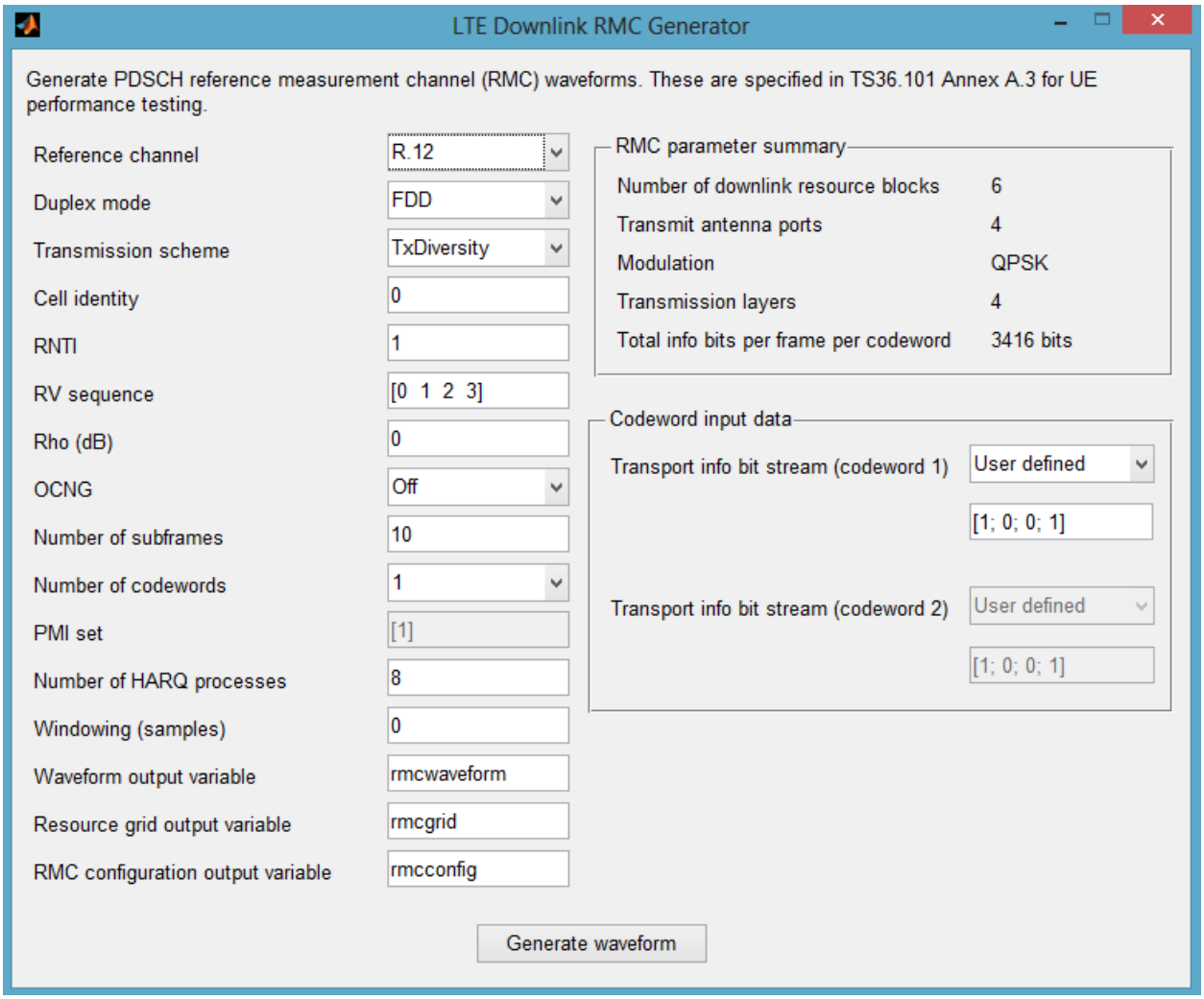


Рис. 10.13. Окно генератора от базовой станции к абоненту

В данном окне задаются параметры генерируемого сигнала на базовой станции, такие как: Количество каналов, вид модуляции, количество обслуживаемых абонентов в секунду, число кодовых слов, вектор инициализации.

На рисунке 19.14 показан вид сгенерированного сигнала, а на рисунке 10.15 трех мерный спектр полученного сигнала.

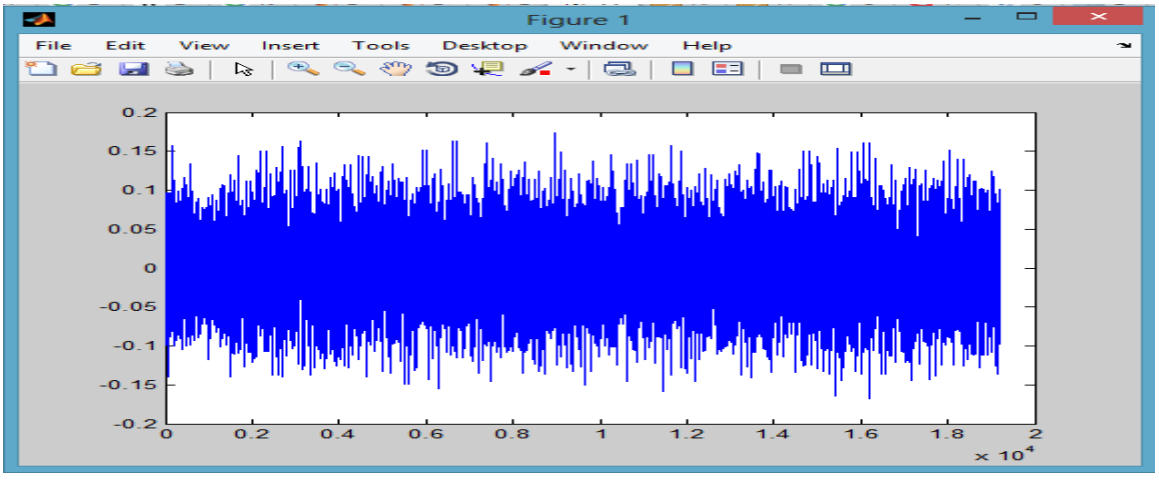


Рисунок 10.14. Сгенерированный сигнал станцией

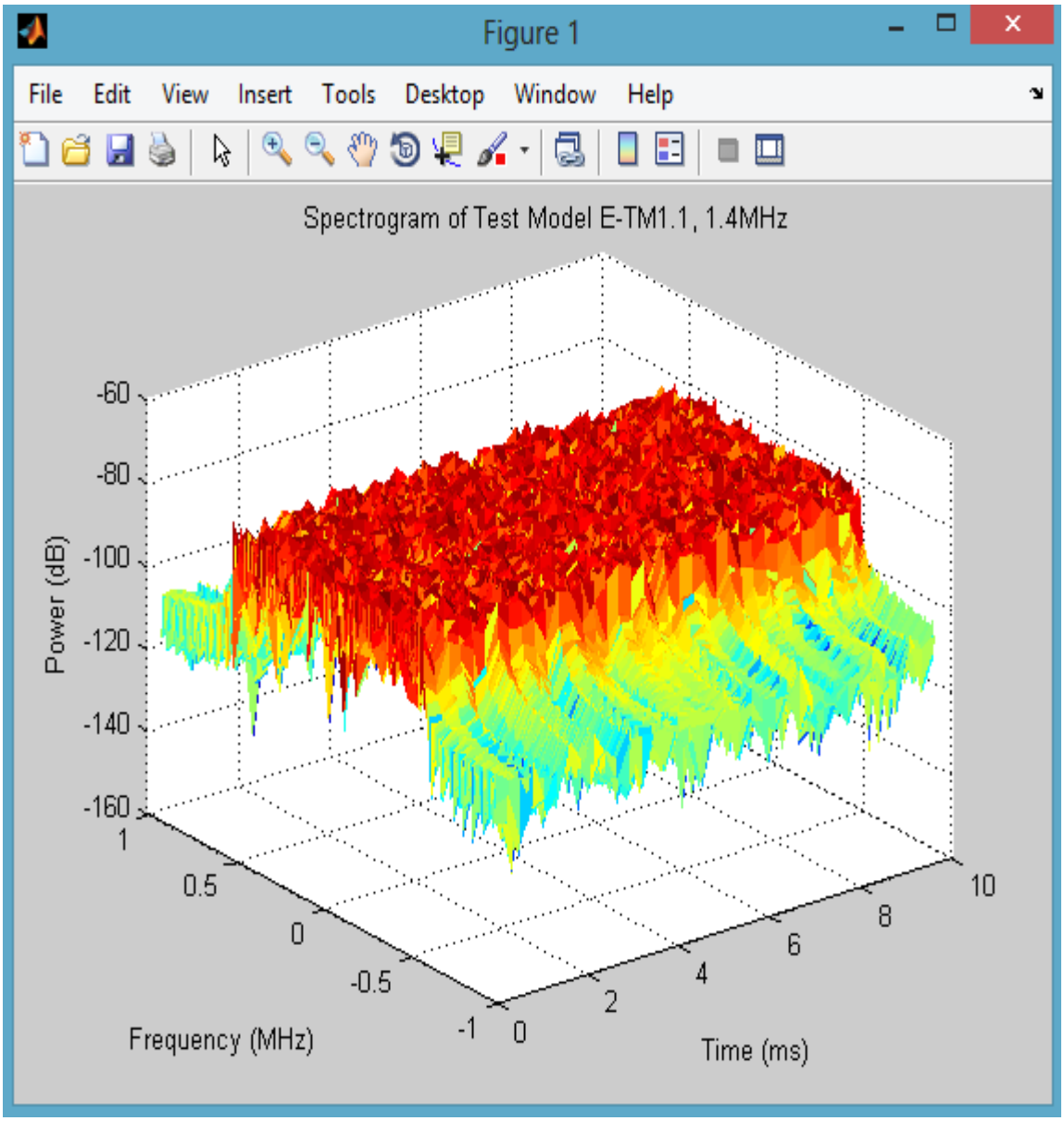


Рис. 10.15. Спектр сгенерированного сигнала станцией в течении 10 секунд

Для генерации тестового сигнала от абонента к базовой станции используется генератор LTE-Uplink RMS Generator.

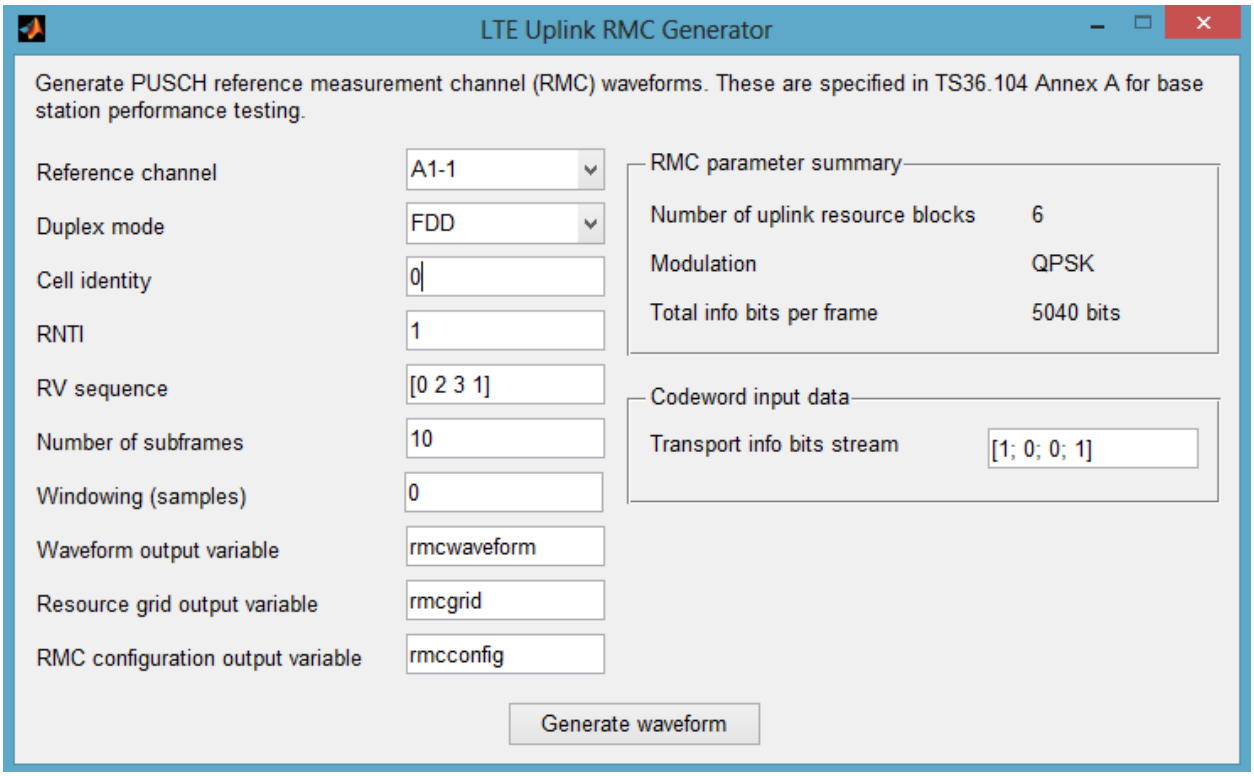


Рис. 10.16. Окно генератора LTE-Uplink RMS Generator

В окне на рисунке 10.16 производятся настройки параметров генерирования от абонента к базовой станции, такие как: Количество каналов для передачи потока пакетов, тип модуляции, вектор инициализации, количество передаваемых пакетов в секунду.

Сигнал, генерируемый генератором LTE-Uplink RMS Generator, представлен на рисунке 10.17.

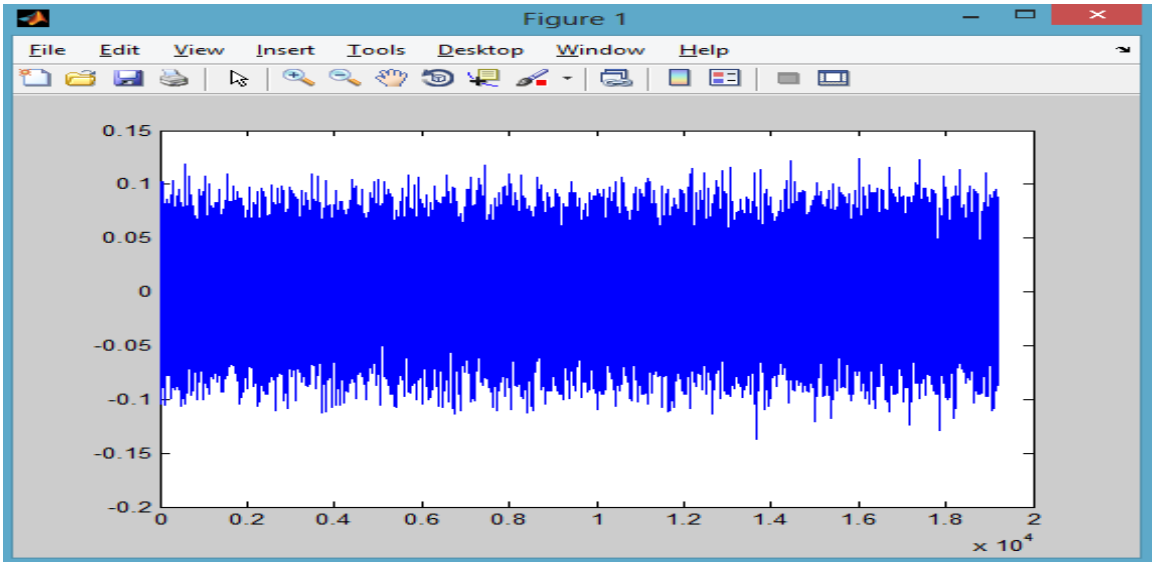


Рис. 10.17. Сигнал генерируемый генератором LTE-Uplink RMS Generator

Для вычисления потерь и пропускной способности системы передачи можно использовать блок LTE PDSCH Conformance Testing.

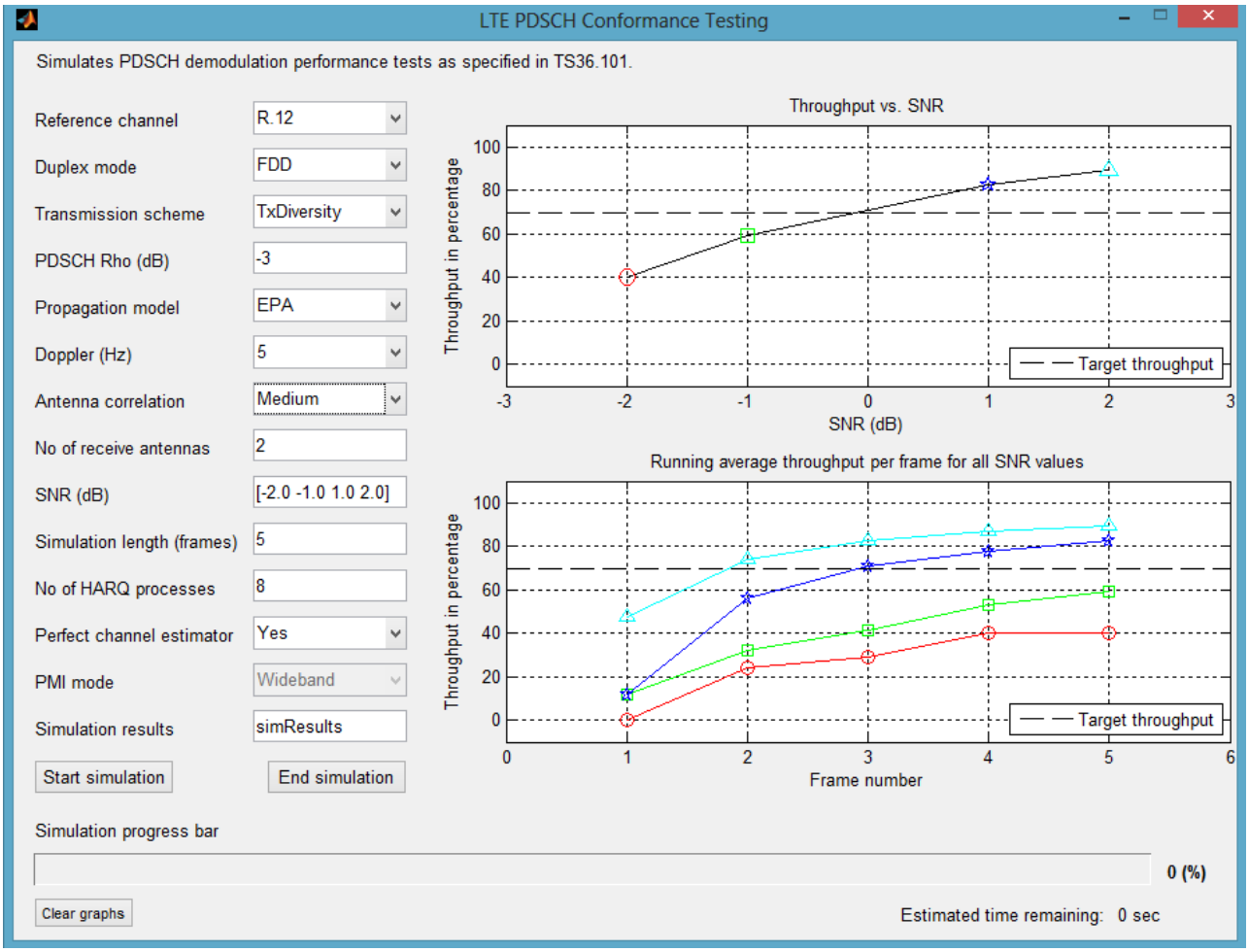


Рис. 10.18. Окно LTE PDSCH Conformance Testing

В данном окне можно произвести настройку параметров линии передачи, таких как: количество каналов, тип модуляции, доплеровскую частоту, уровень шума, настройки антенны и другие. А так же наглядно пронаблюдать изменение количества потерь в линии передачи и пропускную способность системы.

В результате проделанной работы были изучены основные понятия беспроводных сетей LTE, физическая структура построения беспроводной сети LTE. Изучена существующие методы и средства защиты беспроводных сетей LTE. Построена программная структурная схема беспроводной сети LTE среде Matlab с использование встроенного пакета LTE System Toolbox и проведены исследования основных узлов сети LTE.

ЗАКЛЮЧЕНИЕ

Рассмотрены классические шифры, теория классических шифров, компьютерный практикум для классических шифров и задания на самостоятельную работу по классическим шифрам. Преставлены современные шифры с секретным ключом, теория шифров с секретным ключом, компьютерный практикум для шифров с секретным ключом и задания на самостоятельную работу по шифрам с секретным ключо. Рассмотрены отечественные и зарубежные шифры с открытым ключом, теория шифров с открытым ключом, компьютерный практикум для шифров с открытым ключом и задания на самостоятельную работу по шифрам с открытым ключом. Представлены криптографические протоколы в сетях передачи данных и компьютерный практикум для исследования протоколов SSL и TSL. Рассмотрено шифрование в современных системах связи стандартов GSM и LTE и компьютерный практикум для исследования стандарта LTE в MATLAB.

ЛИТЕРАТУРА

1. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии – М.: Гелиос АРВ, 2001. – 480 с.

2. Метлицкий Ю.В. Разработка программного комплекса для визуализации и анализа стандарта криптографической защиты AES, МИФИ 2003 г.

3. Зензин О.С., Иванов М.А. Стандарт криптографической защиты – AES. Конечные поля. – М: КУДИЦ-ОБРАЗ, 2002. -176 с.

4. Б. Шнайер «Прикладная криптография. 2-е издание. Протоколы, алгоритмы и исходные тексты на языке С». - М.: Изд-во "Триумф", 2002. - 816 с.

5. http://www.des-crypto.ru/cryptography/rc4/

6. Асосков А.В. и др. Поточные шифры. - М.: КУДИЦ-ОБРАЗ, 2003. - 336 с.

7. http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html

8. Романец Ю.В. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина. - 2-е изд., перераб. и доп. - М.: Радио и связь, 2001. - 376 с.

9. RSA Laboratories // http://www.rsa.com/rsalabs/node.asp?id=2009

10. Алгоритм RSA : Метод. указания к выполнению лабораторных работ для студентов спец. 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем» очной формы обучения / сост.: О. Н. Жданов, И. А. Лубкин ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2007. – 38 с.

11. Diffie, D. New directions in Cryptography / D. Diffie, M.Hellman // IEEE Transactions on
information theory. November. 1976.
12. Rivest, R. A Method for obtaining digital signatures and public keyCryptosystems / R. Rivest, A. Shamir, L. Adleman // Communications of the ACM. February. 1978.
13. Столингс, В. Криптография и защита сетей: принципы и практика / В. Столингс ; пер. с англ. . – 2-е изд. – М. : Изд. дом «Вильямс», 2001. – 672 с.
14. Коблиц, Н. Курс теории чисел и криптографии / Н. Коблиц ; пер. с англ. М. А. Михайловой и В. Е. Тараканова ; под. ред. А. М. Зубкова. – М. : Науч. изд-во ТВП, 2001. – 254 с.
15. Безопасность на транспортном уровне: SSL и TLS | Лекция | НОУ ИНТУИТ [Электронный ресурс]. – Режим доступа http://www.intuit.ru/studies/courses/553/409/lecture/9387 (дата обращение 13.04.2015).
16. http://matlab.ru/products/LTE-System-Toolbox/lte-system-toolbox_rus_web.pdf (дата запроса 05.04.2016)